FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Deep Learning for the Segmentation of Vessels in Retinal Fundus images and its Interpretation

Pedro Filipe Cavaleiro Breda

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Hélder Filipe Pinto de Oliveira, PhD

Second Supervisor: Ricardo Jorge Terroso de Araújo, MSc

October, 2018

Abstract

Vessel segmentation is a key step for various medical applications, it is widely used in monitoring the disease progression, and evaluation of various ophthalmologic diseases. However, manual vessel segmentation by trained specialists is a repetitive and time-consuming task. In the last two decades, many approaches have been introduced to segment the retinal vessels automatically.

With the more recent advances in the field of neural networks and deep learning, multiple methods have been implemented with focus on the segmentation and delineation of the blood vessels.

Deep Learning methods, such as the Convolutional Neural Networks (CNN), have recently become one of the new trends in the Computer Vision area. Their ability to find strong spatially local correlations in the data at different abstraction levels allows them to learn a set of filters that are useful to correctly segment the data, when given a labeled training set.

In this dissertation, different approaches based on deep learning techniques for the segmentation of retinal blood vessels are studied. Furthermore, in this dissertation are also studied and evaluated the different techniques that have been used for vessel segmentation, based on machine learning (Random Forests and Support vector machine algorithms), and how these can be combined with the deep learning approaches.

Acknowledgements

In first place I would like to thank professor Helder Oliveira, my supervisor, for the guidance and follow-up of the work done and the opportunity to work at INESC TEC. Furthermore, I can't express my gratitude for the support and comprehension regarding all of the problems and setbacks that prevented the conclusion of this dissertation during the required period. His support and comprehension were key to no give up on all the work done over the last months, and for that I will be always thankful.

The biggest thanks to Ricardo Araújo, for being the co-supervisors every student hopes for and even more, and for everything: the help since day one with the review of the literature and research, the willingness to help on every aspect of the work, from providing code samples to the correction and review of this dissertation. Without you, this dissertation would not be half of what it is. To Ricardo, I would also like thank the support and comprehension with situation previously described and apologize of every inconvenient that it may caused to his work.

To my mother, the greatest thanks of all, for all help and the support during this long and difficult journey, in particular these last months. You made unthinkable efforts to make this journey possible, remanding me every step of the way that I had all the capabilities to be whatever I wanted to be. All of your sacrifices allowed me to prose the key aspects for personal growth, which is education, and for that i would be forever grateful.

Also I would like to thank the rest of my family and friends, in special to Ana Beatriz for the support and help dealing with everything in the past couple of months, and to my roommate Pedro Cardão for offering his personal computer setup to run some of the implementations.

Pedro Breda

"Agir, eis a inteligência verdadeira."

Fernando Pessoa

Contents

List of Figures

List of Tables

Abbreviations

2D	Two dimensional
3D	Three dimensional
ACC	Accuracy
AUC	Area under the operating Characteristic curve
CNN	Convolutional Neural Network
FOV	Field of View
FP	False Posite
FN	False Negative
FPR	False Positve Rate
ReLu	Rectified linear unit
RF	Random Forest Classifier
SN	Sensitivity
SP	Specificity
SVM	Support Vector Machines
TP	True Possitive
TN	True Negative
TPR	True Positive Rate

Chapter 1

Introduction

1.1 Context

The retina is a layered tissue coating the interior of the eye responsible for the formation of images, that is, for the sense of vision. By allowing the conversion of light into a neural signal that will be later processed in brain visual cortex, this anatomic structure is one of the most important for the wellbeing of the human individual [1]. As result of its function, the retinal tissue is classified as highly metabolically active, having a double blood supply which allows a direct non-invasive observation of the circulation system [1].

During the last 160 years, retinal imaging has grown rapidly and is now one of the most common practices in clinical care and patient monitoring for people suffering from retinal and/or systemic diseases. The *fundus* photography technique is used in large scale detection of diabetic retinopathy, glaucoma and age-related macular degeneration [2] .

With the more recent advances in the field of neural networks and deep learning, multiple methods have been implemented with focus on the segmentation and delineation of the blood vessels. Each approach attempts to identify and organize the *fundus* image according to a set of features and with that recognize the vessel structure, fovea, macula and the optic disc [3].

The different types of networks, as well as the different types of data available, allows the implementation of a new and improved solution, which can be compared with the previous work done in this area [4].

For this thesis, one goal is to study the result of different combinations of neural networks tested and trained in different databases and do a comparative analysis with other solutions available, considering not only deep approaches for this regard. With this study complete and serving as support, we will attempt to implement a solution, or a set of solutions to automatically segment the retinal blood vessels. The results of the presented solution will then be analyzed, interpreted and benchmarked with the available ones.

1.2 Motivation

Automatic segmentation of vessels in 2D and 3D data is a task that has been already addressed by several researchers. This comes from the importance that these structures have when considering the evaluation of different pathological conditions, such as diabetic retinopathy in retinal *fundus* images, and coronary artery disease in cardiac data. Often, the experts have too much data to analyse, leading to the necessity of having Computer Aided Detection tools that can accurately perform these tasks.

Many methods have been used to perform vessel segmentation. Convolution with matched filters is an important subset of that group. These filters are approximations to the local profiles that vessels are expected to have and their convolution with the data outputs higher values at the locations where the similarity is higher. Even then, it is common that vessels show different types of cross section profiles, making the a priori design of filters a complex task.

Deep Learning methods, such as the Convolutional Neural Networks (CNN), have recently become one of the new trends in the Computer Vision area. Their ability to find strong spatially local correlations in the data at different abstraction levels allows them to learn a set of filters that are useful to correctly segment the data, when given a labeled training set. Furthermore, analyzing the features that the CNN used to solve a task could be useful for humans to understand better the original problem.

1.3 Objectives

The main goal for this dissertation is to study and analyze different approaches based on deep learning techniques for the segmentation of retinal blood vessels. In order to do so, different design and architectures of CNN's will be studied and analyzed, as their results and performance are evaluated and compared with the available algorithms.

One other important objective of this work is to study and evaluate the different techniques that have been used for vessel segmentation, based on machine learning, and how these can be combined with the deep learning approaches: by analyzing the features that the learned models are using to perform classification and combining them with different machine learning techniques, another objective is to propose a solution or set of solutions to perform the retinal vessel segmentation, using this methodology of work.

1.4 Document Structure

This document is divided into seven main sections. The presented section is the introduction to this dissertation. Chapter 2 is a brief description of some pertinent background information regarding the anatomic structure of the retina, its vessel structure and blood supply as well as the *fundus* photography technique. In this chapter both datasets used during the work are also described, detailing all the technical information regarding the image acquisitions and data format.

On chapter 3 the stat of the art of retinal vessel segmentations is presented, the methods studied are divided in unsupervised and supervised, and then subdivided according to the technique used. All of them are presented, analyzed and their results are then summarized and compared in the final of said chapter.

On the chapter 4 is presented the methodology of work, the proposed solution and its detailed implementation is described. The results obtained during the work, its analysis and discussion are presented on Chapter 5. Lastly, on the final chapter (6) are discussed the conclusions and possible future work regarding the developed solution.

Chapter 2

Background

2.1 Anatomic structure of the retina

2.1.1 Retinal layer & structure

The retina is a light-sensitive tissue that exists inside the eye, behaving like a film in a camera. A series of chemical and electrical events occurs within the retina triggered by the optical elements of the eye as they focus an image. The electrical signals resulting from this chain of events are then sent to the brain via nerve fibers, where they are interpreted as visual images [5].

This structure, on human individuals, is located in the inner surface of the posterior two-thirds to three-quarters of the eye [6]. A thin, delicate, transparent sheet of tissue derived from the neuroectoderm, the retina comprises the sensory neurons, which is the beginning of the visual path way. Multiple neurons compose the neural retina (neuroretina). The neural retina is divided into nine layers and is a key component in the production and the transmission of the electrical impulses [7].

The centre of the retina, known as macula, provides the greatest resolving power of the eye, being responsible for the central vision (fine vision), its center is designated as fovea and a representation of this can be observed in figure 2.1. The living tissue of the retina may be imaged using photography, fluorescein angiography, lase polarimetry, or optical coherence tomography [7].

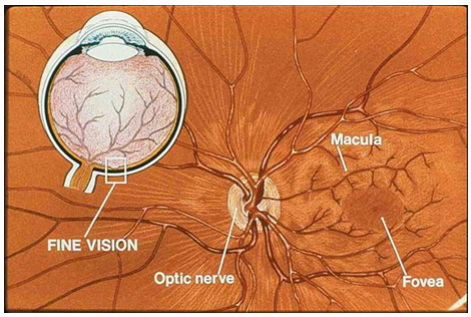


Figure 2.1: Representation of the structures involved in fine vision [7].

2.1.2 Vessels and Blood supply of the retina

To comprehend the function and anatomy of the retina, one key aspect is to study the blood supply and the vessels which constitute it. As stated before, the analysis of these structures is important to diagnose several types of diseases and malfunctions.

Apart from the foveolar avascular zone (FAZ) and the extreme retinal periphery (that can be supplied throughout the choroidal circulation by diffusion since they are extremely thin), the remaining human retina is too thick to be supplied by either one of the retinal or the choroidal circulation alone [8] [9]. Thus, the choroidal circulation supplies the outer retina and the inner retina is supplied by the retinal circulation [10]. The main supplier of blood to the retina is the ophthalmic artery (the first branch of the carotid artery on each side), since both the choroidal and retinal circulation have origin in it [9].

The choroid, a vascular layer of the eye, containing connective tissues, and lying between the retina and the sclera, receives 80% of all ocular blood. The remainder goes to the iris/ciliary body (15%) and the last 5% to the retina [10]. The choroidal circulation is fed by the ophthalmic artery via the medial and lateral posterior ciliary arteries, each of which gives rise to one long and several short posterior ciliary arteries. All the blood in the capillaries of the choroid (choroiocapillaris) is supplied by the short posterior arteries, which enter the posterior globe close to the optic nerve [11], can be seen in figure 2.2.

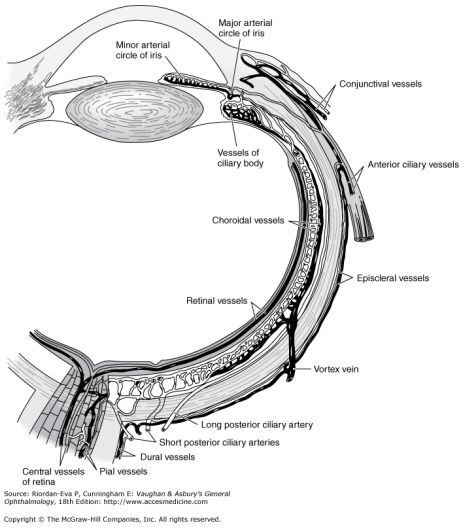


Figure 2.2: Schematic representation of the choroidal circulation [11].

Considering the retinal circulation, the central retinal artery branches off the ophthalmic artery after entering the orbit and enters the optic nerve behind the globe [11]. The central artery subsequently emerges from within the optic nerve cup to give rise to the retinal and inferior circulation with its four main branches, the superior and inferior temporal and nasal retinal arteries [12]. This circulation supplies blood to all the layers of the neuroretina with the exception of the photoreceptors layer (this layer is avascular, so it is dependent on the choriocapillaris for blood). The

retina circulation has a recursive type of layout, being characterized by the temporal retinal vessel curving around the fovea and the FAZ, as displayed in figure 2.3, where these characteristic is noticeable [11] [12].

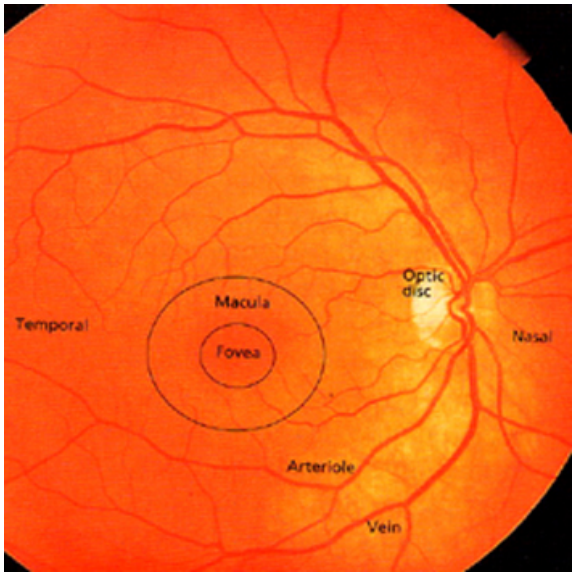


Figure 2.3: *Fundus* Fluorescein Angiography illustrating the retinal circulation [12].

2.2 *Fundus* Image

Ocular *fundus* imaging plays a key role in monitoring the health status of the human eye. Nowadays, the assessment and classification of ocular alterations can be easily made thanks to the large number of imaging modalities. Modalities like photography, fluorescein angiography, lase polarimetry, or optical coherence tomography have evolved as technology is improving [13]. Some of these techniques use contrast when acquiring the images, namely the contrast fluorescein angiography and the indocyanine angiography, whereas photography-based ones do not. Red-free photography, colour *fundus* photography, *fundus* auto fluorescence and infrared reflectance are examples of the latter [14].

Now, the imaging modality which will be used in this thesis - the colour *fundus* photography - is focused. It requires a *fundus* camera, a complex optical system comprising a specialized low power microscope with an attached camera, capable of simultaneously illuminating and imaging the retina. This system is designed to image the interior surface of the eye, which includes the retina, optic disc, macula, and posterior pole [15]. *Fundus* photography is one the most important and older techniques used to obtain these images. It dates back to the early 20th century and has evolved as the time progressed. With origin as a film-based imaging, *fundus* photography was critical to the early developments in diagnoses and pathology studies [14] . With the advent of

digital imaging, the film-based approach became almost obsolete being only used on one-off situations. The use of digital *fundus* imaging allowed to achieve higher resolution, easier manipulation, processing, and tracing of irregularities, and a faster manner of transmitting information [13].

Before working or studying these images, it is important to know which conditions and parameters were used in the image acquisition process, since different cameras produce different results and the images may vary with the resolution, field of vision (FOV) and lighting [14]. An example of the influence of the field of view in the imaging process is showed in figure 2.4, where the retina is photographed using a 40° , 20° and 60° field of view.

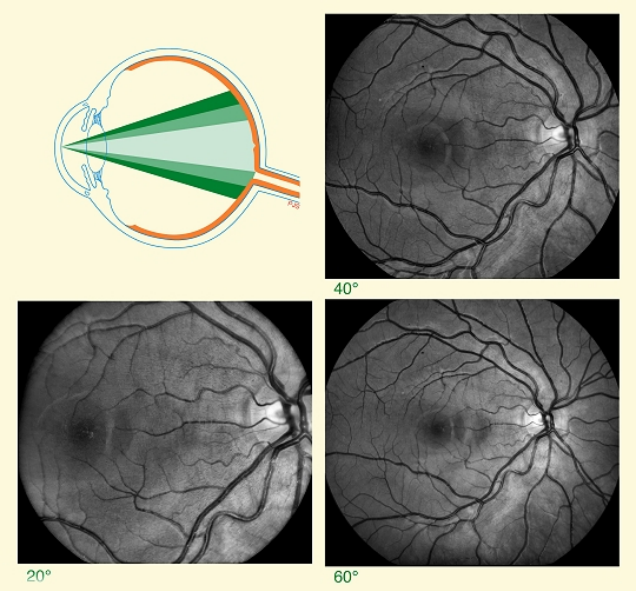


Figure 2.4: Examples of *fundus* photography using 40°, 20° and 60° FOV [13].

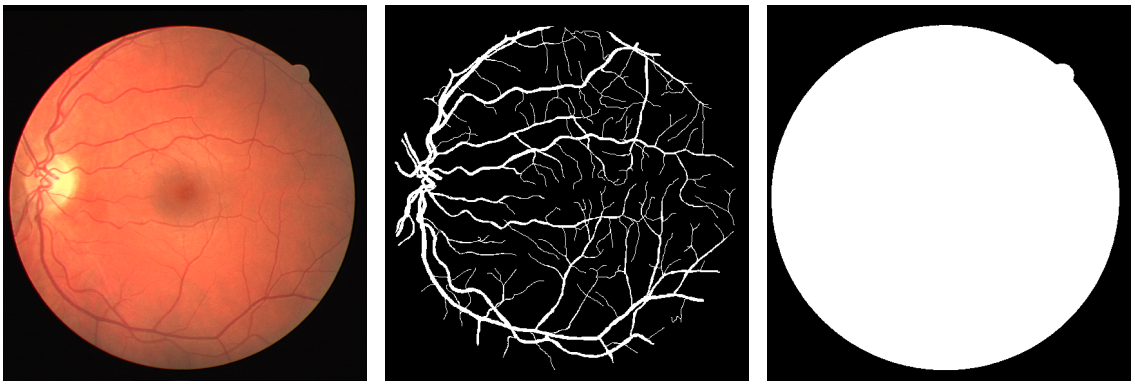
2.3 Databases

Most retinal blood vessels segmentation methodologies are evaluated on either the DRIVE[1], the STARE[2] or on both databases. So, it is important to know and understand the characteristics of each one.

2.3.1 DRIVE Database

Regarding the DRIVE database, the images that compose it are the result of a diabetic retinopathy screening program of 400 diabetic individuals between 25 and 90 years of age, conducted in the Netherlands. From this study, 40 photographs have been randomly selected in which 33 of them do not show any sign of diabetic retinopathy and the rest show some indicators of early diabetic retinopathy.

The images were acquired using a Canon CR5 non-mydriatic 3CCD camera with a 45 degree field of view (FOV). Each image was captured using 8 bits per colour plane at 768 by 584 pixels. The FOV of each image is circular with a diameter of approximately 540 pixels. For this database, the images have been cropped around the FOV. For each image, a mask image delineating the FOV is provided (Figure 2.5c) .

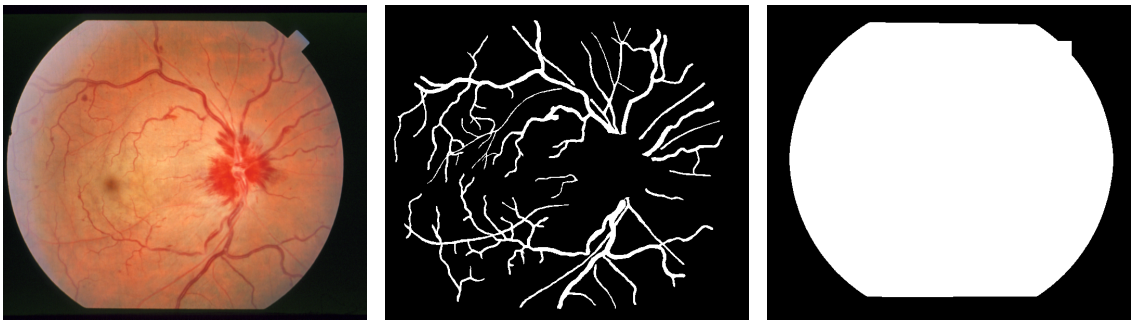


(a) (b) (c)

Figure 2.5: Original DRIVE image (a), it's manual segmentation (b) and binary mask (c).

The database is divided into two sets of images, a training and a testing set, each one of them containing 20 images. Figure 2.5a presents an example training image. A manual segmentation of the vasculature is available for each of the training images and for the test dataset two segmentations are available; one is used as gold standard (Figure 2.5b), the other one can be used to compare computer generated segmentations with those of an independent human observer.

[1]DRIVE Database: https://www.isi.uu.nl/Research/Databases/DRIVE/
[2]STARE Database: http://cecas.clemson.edu/ ahoover/stare/



(a) (b) (c)

Figure 2.6: Original STARE image (a), it's manual segmentation (b) and binary mask (c).

2.3.2 STARE Database

The STARE database is a set of 20 images of retinal fundus photography from which half of them contain pathology. The images were acquired using a TopCon TRV-50 fundus camera at 35 FOV. After the acquisition of the image, they were digitalized to 605x700 pixel, 8 bits per colour Channel. The first half of the dataset comprises images of healthy subjects, while the other half contain pathological cases with abnormalities that overlap with blood vessels, in some cases obscuring them completely.

The presence of pathologies (Figure 2.6a) makes the segmentation more challenging and allows performance evaluation in more realistic conditions. The database contains two sets of manual segmentations prepared by two observers, and the former one is usually considered the ground truth (Figure 2.6b). Similarly to the DRIVE database, for each image the FOV binary delimitation mask was obtained using thresholding (Figure 2.6c).

Chapter 3

State of the art

The vascular network of the human retina is an important diagnostic factor in ophthalmology. An automatic detection and analysis of the vasculature can assist in the implementation of screening programs for the diagnostic of several diseases, computer assisted surgery and biometric identification [3]. The retinal vasculature is composed of arteries and veins appearing as elongated features, which are visible within the retinal image [1].

The segmentation of the vascular network in *fundus* imaging is a nontrivial task due to variable size of the vessels, relative low contrast, and potential presence of pathologies like microaneurysms, haemorrhages, cotton wool spots and bright and dark lesions. Moreover, vessel crossing and branching can further complicate this task [3]. When examining the segmentation process, it is important to consider that the orientation and grey level of a vessel do not change abruptly. Vessels are locally linear, and intensity varies gradually along their length [16]. In addition, retinal vessels are expected to be connected forming a binary treelike structure. However, the shape, size and local grey level of blood vessels can vary hugely, and some background features may show similar attributes [16].

The algorithms for segmentation of blood vessels in medical images are commonly divided into two groups. A first group consisting of the rule-based methods, comprises vessel tracking, matched filter responses, multi-scale approaches and morphology-based techniques [3]. The second group consists of supervised methods (these require manually labelled images for training) which include pattern identification, and classification and recent developments in deep learning [17].

In this Chapter, the segmentation methods which have been applied to retinal vessel segmentation will be briefly described. They will be divided into two classes, one concerning unsupervised methods and other approaches, and another with respect to supervised techniques. After their description, a table summarizing the results is available at the end with all the results from the two classes for a more a complete and detailed comparison.

3.1 Performance Indicators

To evaluate and analyse the performance and the outcome of each segmentation technique and its implementation, it is crucial to comprehend what is involved in the process, which variables are used to define the performance, the way this influences the results and the information that can be withdrawn from it. The outcome in the retinal vessel segmentation process is a pixel-based classification result, meaning that each pixel can be classified either as vessel or surrounding tissue. As a result of this, each pixel, may be seen as belonging to one of the following categories: true positive (TP) when the pixel is identified as a vessel in both the ground truth and segmentation output; true negative (TN) when the pixel is identified as non-vessel in both the ground truth and segmentation output; false positive (FP) when the pixel is identified as vessel in the segmentation output but as non-vessel in the ground truth; false negative (FN) when the pixel is identified as non-vessel in the segmentation output but as vessel in the ground truth. This information is summarized in table 3.1 .

Table 3.1: Vessel classification.

	Vessel present	**Vessel Absent**
Vessel detected	TP	FP
Vessel not detected	FN	TN

Using these indicators, it is possible to calculate some metrics of performance. The true positive rate (TPR) is the fraction of vessel pixels which are correctly identified as belonging to vessels), analogously the false positive rate (FPR) is the ratio of non-vessel pixels which were erroneously classified as belonging to vessels. The ratio of the total number of correctly classified pixels (sum of true positive and true negative) to the number of pixels in the image field of view, is defined as Accuracy (ACC). The ability to detect non-vessel pixels is the specificity (SP) and is the complement of the false positive rate (1-FPR). The sensitivity (SN) reflects the ability of the algorithm to detect the vessel pixels. All this indicators and measurements are transversal to the totality of the references studied and are summarized in table 3.2. Lastly, another important indicator is the value of the area under the receiver operating characteristic (ROC) curve. This curve is the result of the plot of the fraction of pixels correctly classified as vessel (TPR) versus the fraction of non-vessels pixels wrongly classified as vessels (FPR). The closer the curve approaches the top

Table 3.2: Performance indicators and metrics for vessel segmentation.

Measure	**Description**
True Positive Rate (TPR)	True Positive/Vessel pixel count
False Positive Rate (FPR)	False Positive/Non-vessel pixel count
Specificity (SP)	True Negative/(True Negative+False Positive)
Sensitivity (SN)	True Positive/(True positive+False Negative)
Accuracy (ACC)	(True Positive+True Negative)/Field of view pixel count

left corner, the better is the performance of the system. In an optimal system the value of the area under the ROC is 1.

3.2 Unsupervised Learning and other approaches

3.2.1 Unsupervised Learning

The approaches based on unsupervised learning use inherent patterns of blood vessels in the retinal images that can be used to determine that a particular pixel belongs to the vessel structure or not. In these methodologies, the ground truth does not contribute directly to the algorithm design [3].

The methods of work classified as unsupervised learning are based on a specific technique approach (such as match filtering, morphological processing, vessel tracing/Tracking or multi-scale approach) or on the conjugation of different approaches. Several methodologies have been proposed over the past years for the segmentation of retinal blood vessels. One of the first examples of these was proposed by Salem *et al.* [18], where the authors presented a Radius Based Clustering Algorithm, a method using a distance-based principle to map the distributions of the image pixels before performing segmentation [18]. The authors considered three features: the green channel intensity, the local maxima of the gradient magnitude, and the local maxima of the largest eigenvalue calculated form the Hessian matrix. The results shown that, in comparison with a k-Nearest Neighbors classifier, the proposed implementation is better in the detection of small vessels [18].

Serving as example of another techniques used, Kande *et al.* [19] presented a fuzzy based implementation, where the non-uniform illumination of the image can be corrected by using the information of the intensity of the green and red channels. The matched filtering technique was used to enhance the contrast of the vessel against the image background. After this, a spatially weighted fuzzy C-mean clustering and a connected component labeling are used to identify the vessel [19].

And the method proposed by Villalobos *et al.* [20], that using this and with information of the surrounding elements the segmentation process is possible, considering that pixels with gray-levels above the threshold are assigned to the vessel structure, and those equal to or below the threshold are assigned to the background. In the following subsections is presented the most relevant work based on the different techniques previously mentioned.

3.2.2 Matched Filtering

Matched filtering is an effective method for retinal vessel enhancement, consisting on the convolution of a 2D kernel with the retinal image. The kernel is designed to resemble the pattern we are looking for in the image, at an unknown position and orientation. The matched filter response indicates the presence of the feature. The first method using the matched filter approach was proposed by Chaudhuri *et al.* [21]. It was based on the statement that the grey distribution of the vascular profile is consistent with the Gaussian properties and that the image can be convoluted with a filter to extract the target object [21]. Hoover *et al.* [22] proposed the use of a threshold descent search

algorithm to extract the blood vessels after matched filtering. The algorithm considers the local features of the retinal vessels and the regional characteristics of the vascular network distribution simultaneously. By doing so, the proposed algorithm can greatly reduce the error, at the cost of complicating the overall calculation [22].

Recently, different researchers have improved the method of matched filtering, example of this is the approach presented by Zhang *et al.* [23]. The authors applied a model considering double sided thresholding to improve the result of matched filtering, achieving reduced false positives in pathological images. After this, the same authors also proposed a method which used the matched filtering and the first-order derivative of the Gaussian filter to identify the vascular boundaries and improve the accuracy of vascular segmentation [23].

More recently, Li *et al.* [24] suggested a method for vessel extraction using multi-scale adaptations of the matched filter. This allowed to enhance the contrast of the image while suppressing the noise, before employing a double thresholding method to detect the vessels [24].

The result of the matched filter algorithm depends on the degree of matching between the template and the blood vessel, which can be affected by different factors such as central light reflection, radius change, noise interference and lesion interference [17].

3.2.3 Morphological Processing

Morphological image processing is a collection of techniques for digital image processing based on mathematical morphology and operations [3]. These are essential in the field of image processing and they are useful in the representation and description of region shapes such as features, boundaries and skeletons. Several methods for the segmentation of retinal vessels using this approach have been developed. Zana *et al.* [25] proposed a method to extract the vascular tree using mathematical morphology, considering characteristics of retinal vessels such as connectivity and local linear distribution. This implementation showed some vulnerabilities since it was too dependent on structural elements. With the previous method serving as a base, Ayala *et al.* [26] proposed a new method that defines the averages of a given fuzzy set by using, different definition of mean of a random compact set [26].

Mendonça and Campilho [27] used multi-scale top hat transformation to enhance the retinal blood vessels combined with the extracted vascular centerlines to obtain the vessel segmentation. In order to improve this implementation, M. M. Fraz *et al.* [15] combined the vessel centerline detection with the morphological bit plane slicing to achieve the final segmentation. One approach showing very positive results (an accuracy of 96%) was proposed by Miri *et al.* [28], where the curvelet transform and multi-structural element morphological reconstruction are used to process the *fundus* image. Lastly, Soares *et al.* [29] resorted to morphological reconstruction and a regional growth method before using a Gabor wavelet transformation to extract the blood vessels.

The presented methodologies have the advantage of suppressing noise and being fast and efficient. However, considering that the segmentation is seriously dependent on the selection of structural elements and these methods do not make full use of vascular characteristics, they are normally combined with other approaches for the segmentation task [17].

3.2.4 Vessel Tracing/Tracking

Vessel tracking algorithms segment a vessel between two points using local information and work at the level of a single vessel rather than the entire vasculature. Vessel tracking follows vessel centerlines guided by local information, usually trying to find the path that better matches a vessel profile model [3].

Several implementations have been proposed using this approach, one of the first dates back to 1999, when Can *et al.* [30] described a real time algorithm which is based on recursively tracking the vessel starting from the initial seed-points using directional templates. Each directional template is designed to give the maximum response for an edge oriented in a particular direction [30]. The algorithm takes a step in the direction of maximum response and this procedure is repeated at the new point until the segmentation process is completed [30]. Other approach based on multi-scale line-tracking and post-processing with morphological operations was proposed by Vlachos *et al.* [31].

An approach that uses the maximum a posteriori probability criterion to perform the segmentation based on the vascular diameter was initially presented by Mouloud Adel *et al.* [32] and later improved by Yin *et al.* [33]. The proposed improvements include the use of a semi-ellipse as the dynamic search window in order to obtain the local grey level statistic information [33]. An algorithm using a parametric model which exploits geometrical properties for parameter definitions was proposed by Delibasis *et al.* [34]. This uses a measure of match to define the similarity between the model and the given image. This implementation was tested on the DRIVE database with very positive results as is demonstrated in Table 3.6. Although the vessel tracking/tracing-based methodologies are simple, intuitive and fairly easy to understand, there are some problems associated to them. The algorithms based on these methodologies are unable to detect vessels or vessel segments which have no seed points and can miss some bifurcation points resulting in undetected sub-trees. To improve their performance, these algorithms are commonly used in conjunction with matched filtering and morphological operations [17].

3.2.5 Multi-scale approaches

Some of the methods mentioned before used multi-scale approaches and are based on the principle that a vessel is defined as a contrasted pattern with a Gaussian like shape cross-section profile, piece-wise connected, and locally linear, with a gradually decreasing diameter. This idea was established since the width of a vessel decreases as it travels radially outward from the optic disk and such change is a gradual one [3].

The idea behind scale-space representation for vessel segmentation is to naturally represent and characterize vessels of varying widths. Frangi *et al* [35] proposed a method that examined the multi-scale second order local structure of an image (Hessian) in the context of developing a vessel enhancement filter [35]. The analysis of the eigenvalues of the Hessian matrix allows to find the principal directions in which the local second order structure of the image can be decomposed, and consequently detect the direction of small intensity curvature along a vessel.

Other implementation using multi-scale approaches was presented by Martinez-Perez *et al.* [36] where the size, orientation, and width of vessels are obtained through two geometrical features accounting for the first and second derivatives of the intensity, thus allowing to infer about the topology of the image.

The main advantage of using this methodology is the fact that there is a framework where vessels of different sizes are easily segmented. The results yielded form these methodologies presented an improvement in the accuracy and area under the ROC curve, as can be observed in table 3.6, when compared with some of the previously presented unsupervised implementations.

3.3 Supervised methods

The supervised methods exploit some prior labeling information to decide whether a pixel belongs to a vessel or not. The rule for vessel extraction is learned by the algorithm and is based on a training set of reference images, previously processed and manually segmented with precision by an ophthalmologist or a related professional. Generally, these images are referenced as the ground truth or golden rule. As a result, the availability of a ground truth is a critical requirement for every implementation of this kind, making these approaches not easy to implement in many real-life applications.

The methods based on this type of learning mechanism can be divided into two mainly approaches: the ones based on traditional machine learning and the ones based on deep learning. The approaches based on traditional machine learning use a set of features to learn and then produce an output, whereas the approaches based on deep learning network perform automatic feature extraction without human intervention and with that these algorithm can be trained using labeled data [37].

A deep-learning network trained on labeled data can then be applied to unstructured data, giving it access to much more input than traditional machine-learning based approaches, and by doing so allowing a higher performance, since the more data a net can train on, the more accurate it is likely to be [38].

Although the approaches based on traditional machine learning are the most commonly used, in the recent years the use of deep learning approaches as risen and these produced some of best results (regarding accuracy and area under the ROC curve, as can be observed in table 3.6) in the task of retinal vessel segmentation [39].

3.3.1 Traditional Machine learning

Regarding traditional machine learning, the first approach was proposed by Staal *et al.* [40]. The authors presented a ridge-based vessel segmentation methodology from coloured images that rest on the intrinsic property that vessels are elongated structures. The system is based on the extraction of image ridges, which approximately coincide with the vessel centerlines. The ridges are detected using the green channel of the *fundus* image (RGB), as it is the channel where the contrast between vessel and background is higher. The use of ridges allows the composition of primitives in the

form of line elements [40]. The extracted ridges allow to form primitives with the shape of line elements, such that the image may be partitioned into patches by assigning each image pixel to the closest line element from these sets. Every line element constitutes a local coordinate frame for its corresponding patch in which local features are extracted for every pixel. The feature vectors are classified using a k-Nearest Neighbours classifier and sequential forward feature selection [40].

The second implementation analysed was proposed by Soares *et al.* [29]. This approach uses a 2D Gabor wavelet and supervised classification to segment the blood vessels of the retina. In this method, each pixel is represented by a feature vector including measurements of the 2D wavelet taken from different scales. The resulting feature space is used to classify each pixel as either a vessel or non-vessel pixel. The classification is done by applying a Bayesian classifier with class-conditional probability density function described as Gaussian mixtures, yielding a fast classification, while being able to model complex decision surfaces [29]. An important fact to consider about this implementation is that it do not work very well with non-uniform illumination as it occasionally produces false detections on the border of the optic disc, haemorrhages and other types of pathologies that present strong contrast with the background [29].

The application of line operators and Support Vector Machine for pixel classification is proposed by Ricci *et al.* [41]. In this implementation, the retinal blood vessel segmentation is based on features vectors built from line operators. A line detector (which is based on the evaluation of the average grey level along the lines of fixed length passing through the target pixel at different orientations) is applied to the green channel of the retinal image, and the classifier output is thresholded to obtain a pixel classification as shown in figure 3.1.

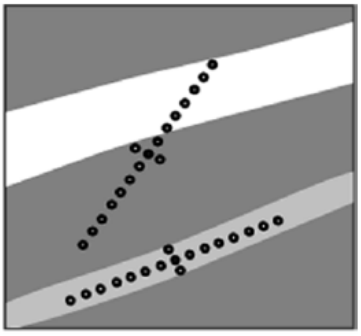


Figure 3.1: Example of a line detector with its orthogonal line as implemented by Ricci *et al.* [41].

The use of local differential computation of the line strength makes the line detector robust to non-uniform illumination and contrast providing more interesting results then the previously methods analysed. This implementation requires fewer features and as result the computational effort for feature extraction is considerably less, consequently the training time is reduced when compared with another implementation [41].

Marin *et al.* [42] presented an implementation based on a neural network. The approach uses a neural network scheme for pixel classification and computes a 7D vector composed of gray-level

and moment invariants-based features for pixel representation. The neural network used for the training and classification was a multilayer feed forward network. The input layer consists of seven neurons, following this, three hidden layers composed of fifteen neurons each were used. Lastly the output layer was comprised of a single neuron. A non-linear logistic sigmoid activation function was used in all neurons, such that the output lies between 0 and 1. The input images were monochromatic and obtained by extracting the green channel from the original retinal image. This methodology, as others based on neural networks, proves to be effective and robust with different image conditions and in multiple image databases, even when the neural network is trained using a single database [42].

Lastly, M. M. Fraz *et al.* [43] presented a method for segmenting blood vessels by using an ensemble classifier of boosted and bagged decision trees. This implementation uses a 9D feature vector which includes the orientation analysis of the gradient vector field (one feature) for removal of bright and dark lesion with the vessel enhancement, morphologic transformation (one feature) for eradication of bright lesions, line strength measures (two features), and a Gabor filter response at multiple scales (four features) for eliminating the dark lesions. The last features were extracted from the intensity of each pixel in the inverted green channel of the retinal image [15]. Regarding the classification, this was based on boot strapped and boosted decision trees, a classic ensemble classifier which has been used in several applications of different fields [17].

3.3.2 Deep Learning

In recent years, methods based on deep learning, which is an important branch of machine learning, have achieved remarkable results in target classification and segmentation tasks in the field of computer vision. Some of these methods have been applied to the task of segmentation of retinal blood vessels. In tasks such as image classification and segmentation are used convolutinal neural newtorks (CNN), alternating convolutional and pooling layers, and finally an output layer (commonly a Fully connected layer).

In the case of traditional hand-designed features, they must be carefully thought and designed before classification, making such process very tedious, time consuming, and sometimes difficult for non-experts. Another crucial aspect to consider when working with hand-designed features, is that these must be redesigned for datasets with different characteristics. In contrast, the use of CNN allows to extract features from raw images, avoiding the use of hand-designed features [44].

The process of feature extraction is the core concept of a convolutional neural network as this type of network relies on the principle of convolution to extract information at multiple levels using both the convolutional and the Pooling layers to do so. With local receptive fields, neurons in a CNN can detect elementary visual features such as oriented edges, end-points and corners. These are then combined by the successive layers in order to capture higher-order features [45] [46].

Considering l a convolutional layer, with the input that consists of $m_1^{(l-1)}$ feature maps from the previous layer, each of whom with dimension $m_2^{(l-1)}$ x $m_3^{(l-1)}$ (in the particular case of l=1 the input of said layer is an image), the i^{th} feature map that results from the convolution process is given by 3.1 where $B_i^{(l)}$ is a bias matrix and $K_{i,j}^{l}$ is the filter used in the convolution, with size $2h_1^{(l)}$

+ 1 x $2h_2^{(l)}$ + 1 that connects the j^{th} feature map in layer (l-1) with the i^{th} feature map in layer l. In this generic case the output of l consists of $m_1^{(l)}$ feature maps of size $m_2^{(l)}$ x $m_3^{(l)}$.

$$Y_i^{(l)} = B_i^{(l)} + \sum_{j=1}^{m_1^{(l-1)}} K_{i,j}^{(l)} \circledast Y_j^{(l-1)} \tag{3.1}$$

Regarding the pooling operations, these are used in convolutional neural networks to make the detection of certain features in the input invariant to scale and orientation changes. This type of layers generalize over lower level and more complex information. Apart to help over-fitting by providing an abstracted form of the representation, this also reduces the computational cost by reducing the number of parameters to learn and provides basic translation invariance to the internal representation [47].

In general, pooling operates by placing windows at non-overlapping positions in each feature map and keeping one value per window such that the feature maps are sub sampled. Pooling acts as a generalizer of the lower level information and so enables us to move from high resolution data to lower resolution information [46]. Considering now l as a pooling layer, more specific a Max pooling one, the input of said layer comprises $m_1^{(l-1)}$ feature maps resulting from the previous layer. The output of each neuron $u_{i,j}^{(l)}$ that constitutes the feature map $Y_i^{(l)}$ resulting form l is given by the set of equations 3.2.

$$\begin{aligned}
u_{i,j}^{(l)} &= max(u_{i-1,j-1}^{(l-1)}, u_{i,j-1}^{(l-1)}, u_{i-1,j-1}^{(l-1)}) \\
u_{i,j+1}^{(l)} &= max(u_{i-2,j-2}^{(l-1)}, u_{i,j-2}^{(l-1)}, u_{i-2,j-2}^{(l-2)}) \\
u_{i+1,j}^{(l)} &= max(u_{i-3,j-3}^{(l-1)}, u_{i,j-3}^{(l-1)}, u_{i-3,j-3}^{(l-3)}) \\
u_{i+1,j+1}^{(l)} &= max(u_{i-4,j-4}^{(l-1)}, u_{i,j-4}^{(l-1)}, u_{i-4,j-4}^{(l-4)})
\end{aligned} \tag{3.2}$$

Regarding now the state of the art in solutions based on deep learning, one of the first was proposed by Wang *et al.* [44]. This supervised method for blood vessel segmentation uses a Convolutional Neural Network (CNN) as a hierarchical feature extractor, and an ensemble of Random Forests which outputs a classification according to those features [44].

The authors [44] used a 6-layer architecture as represented in figure 3.2. The constitution of each layer, the kernel size and the dimension of the learned features is presented on Table 3.3.

Regarding the classification process, an ensemble of random forest models is used, and a winner-takes-it-all strategy is followed. The learned features by the CNN (S2, S4, and C5, as showed in figure 3.2 and present in table 3.3) are fed into independent random forest classifiers.

The classifiers work in parallel and the results are then combined by a winner-takes-all ensemble to produce the final classification. The winner-takes-all classifier is a simple procedure where the class output is determined by the base classifier which achieves the best classification performance among all the classifiers. A detailed representation of the entire classification process is available in figure 3.3. The results of the segmentation process are interesting relative to

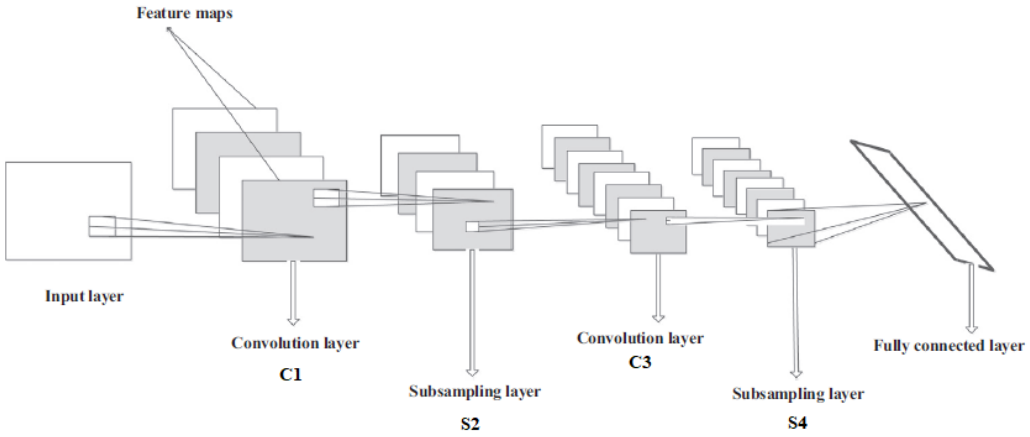


Figure 3.2: Schematic representation of the Convolutional Neural Network presented by *Wang et al.* [44].

Table 3.3: Architecture of the implemented Convolutional Neural Network by *Wang et al.* [44].

Layer	Type	Maps and Neurons	Kernel Size	Stride	Padding	Reference
0	Input	1 Map of 25 x 25 neurons				
1	Convolutional	12 Maps of 22 x 22 neurons	4 x 4	1	1	C1
2	Subsampling	12 Maps of 11 x 11 neurons	2 x 2	2	0	S2
3	Convolutional	12 Maps of 8 x 8 neurons	4 x 4	1	1	C3
4	Subsampling	12 Maps of 4 x 4 neurons	2 x 2	2	0	S4
5	Convolutional	100 Neurons	4 x 4	1	1	C5
6	Output	1 neurons				

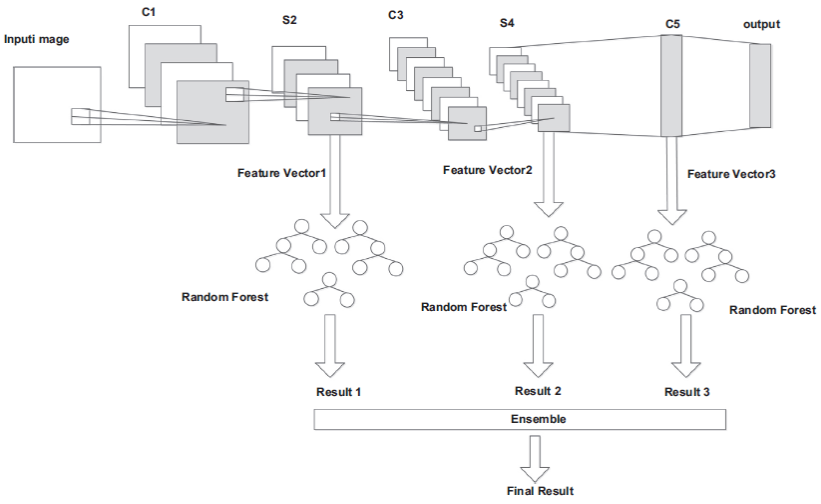


Figure 3.3: Structure of the proposed method by Wang *et al. [44].*

the accuracy in comparison with some of the unsupervised methodologies, an example is showed in figure 3.4.

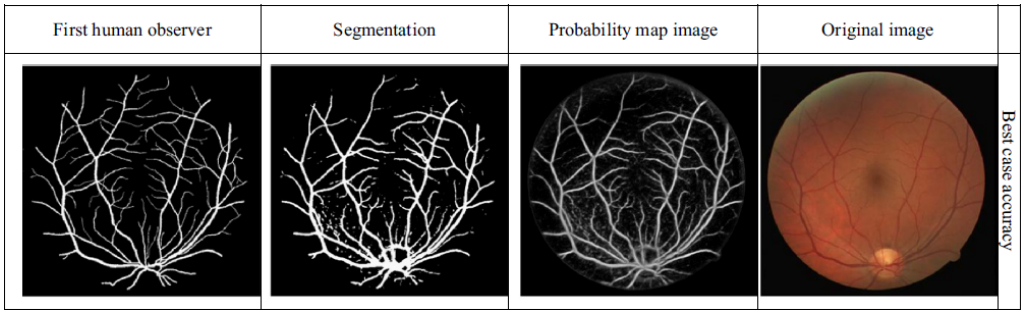


Figure 3.4: Best accuracy results obtained by *Wang et al.* [44] on DRIVE image.

Melinscak *et al.* [48] performed retinal vessel segmentation using a CNN having max-pooling layers instead of subsampling or down-sampling. The layers of this CNN consist of a sequence of convolutional, max-pooling and fully connected layer. By using max-pooling it is possible to map input samples into output class probabilities using several hierarchical layers to classify extracted features [48] . This implementation is based on a CNN with 10 layers, whose architecture is presented on table 3.4. The first layer is the input layer, then a bank of 2D filters is applied in each

Table 3.4: Architecture of the implemented Convolutional Neural Network by *Melinscak et al.* [48].

Layer	**Type**	**Maps and Neurons**	**Kernel size**	**Stride**	**Padding**
0	Input	1 Map of 65x65 Neurons	-	-	-
1	Convolutional	48 Maps of 60x60 Neurons	6x6	1	0
2	Max-Pooling	48 Maps of 30x30 Neurons	2x2	1	0
3	Convolutional	48 Maps of 26x26 Neurons	5x5	2	2
4	Max-Pooling	48 Maps of 13x13 Neurons	2x2	1	0
5	Convolutional	48 Maps of 10x10 Neurons	4x4	2	0
6	Max-Pooling	48Maps of 5x5 Neurons	2x2	1	0
7	Convolutional	48Maps of 4x4 Neurons	2x2	2	0
8	Max-Pooling	48Maps of 2x2 Neurons	2x2	1	0
9	Fully connected	100 Neurons	1x1	-	-
10	Fully connected	2 Neurons	1x1	-	-

convolutional layer in order to obtain maps of features. Max-Pooling layers are fixed and take square blocks of Convolutional layers and reduce their output into a single feature. The selected feature is the most promising as Max-Pooling is carried out over the block. Following this, there are alternating steps of convolutional and Max-Pooling layers.

Finally, the last two layers are fully connected and they are used to further combine the previous outputs, creating a 1D vector of features. The second fully connected layer has two neurons, one for each class, since the classification is binary (vessel or non-vessel). The final output is the probability of each pixel been being vessel [48].

This implementation was tested and trained on the DRIVE database and the most relevant results are presented on figure 3.5 (in this particular case, the binary result of the segmentation

wasn't provided in the published paper, instead the presented image is result of the final softmax activation).

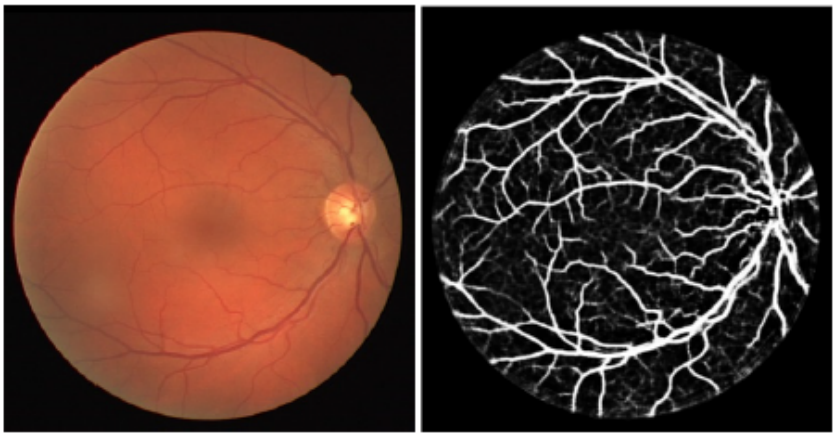


Figure 3.5: Best result of the best AUC score and Softmax activation obtained by Melinscak *et al. [48].*

Fu *et al.* [49] presented a segmentation method somehow similar to the work proposed by Melinscak *et al.* [48]. Contrasting with the methodology proposed by Melinscak *et al.* [48], Fu *et al.* [49] uses an image-to-image training system providing a multi-scale and multi-level visual response. This method uses a Fully Convolutional Neural Network to learn the discriminative features and generate a vessel probability map. Afterwards, a fully connected Conditional Random Field (CRF) accounts for global pixel correlation and produces a binary vessel segmentation result.

Distinct from the method proposed by Melinscak *et al.* [48] that performs as a pixel-to-pixel classification, this proposed method treats the vessel segmentation as a contour detection problem. To tackle this, the Fully Convolutional Neural Network was built on top of a holistically-nested edge detection (HED) [50]. HED automatically learns rich hierarchical representations (guided by deep supervision on side responses) that are important to solve the challenging ambiguity in edge and object boundary detection. An example of a HED implementation is visible on the figure 3.6 [50].

Considering this, Fu *et al.* [49] propose a four-stage implementation, in which each stage includes multiple convolutional and ReLu (Rectified Linear Unit) Layers. The side-output layers are connected to the last convolutional layer in each stage to deem the deep layer supervision [49]. The Conditional Random Fields are used to produce coarse maps at pixel-level segmentation and to suppress the lack of smoothness of the fully CNNs. The figure 3.7 illustrates the presented approach.

One of the most interesting methods for retinal blood vessel segmentation using deep learning was proposed by Liskowski and Krawiec [51]. This method uses a deep neural network (CNN) trained on a large sample of examples pre-processed with global contrast normalization, and zero-phase whitening. Augmentation through geometric transformations and gamma correction was

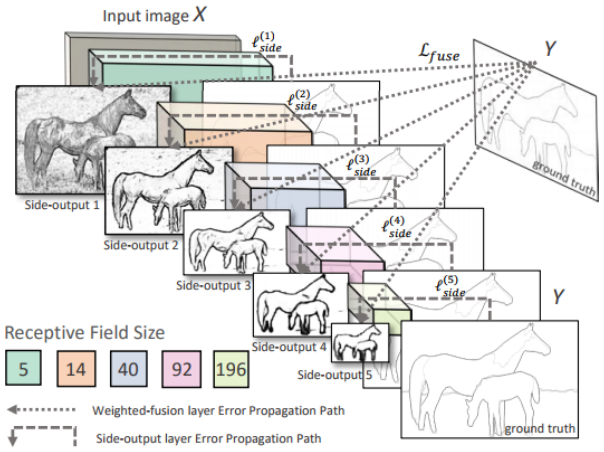


Figure 3.6: Network architecture for edge detection using HED [50].

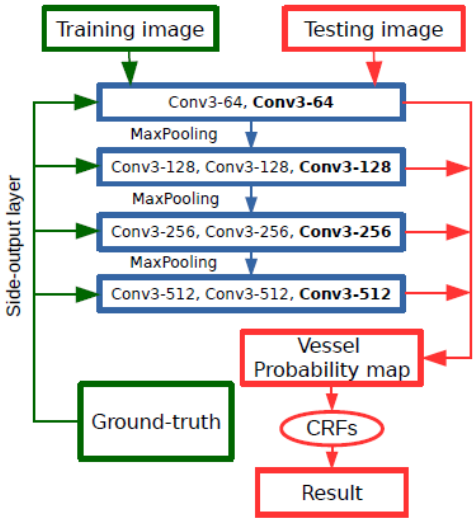


Figure 3.7: Structure of the proposed method by Fu *et al* [49].

performed at training time [51].

The authors studied several variants of CNN architectures, such as including or not max-pool layers and the addition of structured prediction (where a network classifies multiple pixels simultaneously). This study uses three different databases (DRIVE, STARE and CHASE) to train, cross-train and test the all the architectures implemented. By doing so, this study is one of the most complete and detailed of all methodologies based on deep learning in the field of retinal blood vessel segmentation.

The training process starts with the extraction of m-by-m patches (m was set to 27 on their

pipeline). Afterwards, image processing is conducted, by means of global contrast normalization and zero-phase whitening. Even though deep learning architectures can learn from raw images, the authors experiments shown that such pre-processing increased the performance of the entire pipeline. For each training *fundus* image, 20 000 patches like this were generated at random, such that the training set has 400 000 patches in DRIVE and 380 000 in STARE [51]. The most relevant and noteworthy architectures studied by the authors are shown in table 3.5.

Table 3.5: Two most relevant architectures proposed by Liskowski and Krawiec [51].

Implementation	**Layer**	**Maps**	**Kernel size**	**Stride**	**Padding**
Plain	Convolutional	64	4 x 4	1	0
	Convolutional	64	3 x 3	1	1
	Max-Pooling	-	2 x 2	2	0
	Convolutional	128	3 x 3	1	1
	Convolutional	128	3 x 3	1	1
	Max-Pooling	-	2 x 2	2	0
	Fully Connected	512	-	-	-
	Fully Connected	512	-	-	-
	Fully Connected	2	-	-	-
No-POOL	Convolutional	64	3 x 3	1	1
	Convolutional	64	3 x 3	1	1
	Convolutional	128	3 x 3	1	1
	Convolutional	128	3 x 3	1	1
	Fully Connected	512	-	-	-
	Fully Connected	512	-	-	-
	Fully Connected	2	-	-	-

The architectures presented on table 3.5 only differ in spatial pooling. The Max-Pooling process is performed over a 2 x 2 pixels window with a stride of 2 and is only used in the Plain implementation [51]. The No-POOL implementation does not use Max-pooling, since recent results showed that networks which do not further down-sample the features might perform better when applied to small images [52].

Apart from these implementations, the authors considered four others, all using the Plain network architecture: GCN, ZCA, AUGMENTED and BALANCED. All are composed of a stack of convolutional layers that are followed by three Fully Connected layers (Table 3.5). The main goal of these implementations was to analyse the performance of the Plain architecture when trained on data pre-processed by the methods previously described (Global Contrast Normalization, Zero-phase whitening and the image augmentations) and a on balanced data (with equal proportion of decision classes). An example of the data used in the training process is visible on the figures 3.9-3.10.

The different implementations were tested in both DRIVE and STARE databases with interesting results. Considering the overall experiment, the results outperform all of the previous algorithms published, relative to the most commons performance indicator, such as accuracy with average value superior to 0.97 on the STARE and area under the ROC curve averaging 0.99 on the

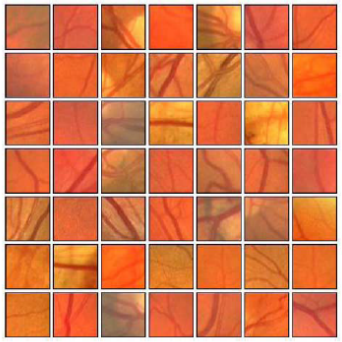


Figure 3.8: Example of Training patches after applying GCN transformation [51].

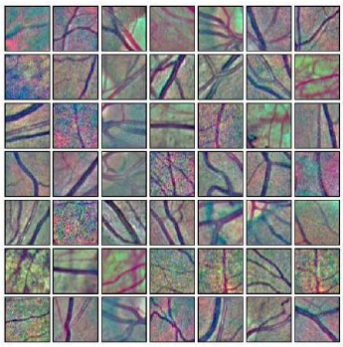


Figure 3.9: Example of Training patches after applying ZCA whitening transformation [51].

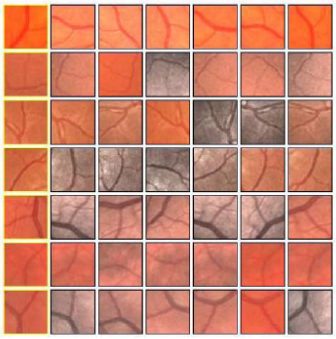
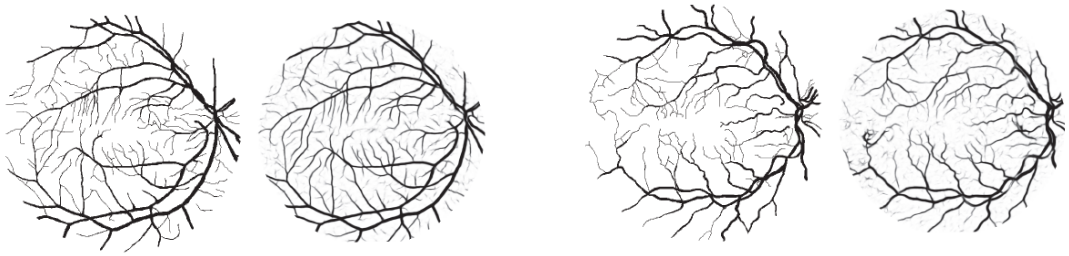


Figure 3.10: Example of Training patches using data augmentation [51].

STARE. The results also show that the proposed method is sensitive in the detection of fine vessel and fares well with pathological cases.

From the six implementations explained before, the best performance was achieved by the network trained on balanced data, obtaining the best-to-date results on both databases (considering the area under the ROC curve). The second best-performing architecture is the No-Pool. The two best performing architectures do not have pooling layers, what supports that down-sampling may be unworthy when small patches are being classified. Considering this, these layers may be discarded and by doing so, the dimensionality of the patches is kept across the entire network, allowing convolutional filters in successive layers to be applied at the same resolution.

Following this, Liskowski and Krawiec [51], using the two best architectures as base, also implemented structured prediction model. The addition of this model resulted on an improvement of the previously presented results increasing the AUC by 0.5% on DRIVE and 1.1% on STARE. In figure 3.11, the best segmentations of this architecture in DRIVE (a) and STARE (b) databases are presented.



(a) (b)

Figure 3.11: Ground truth (left) and segmentation result (right) for two healthy subjects: (a) DRIVE and (b) STARE [51].

3.4 Summary and Comparison

This chapter presented some of the most relevant approaches regarding retinal blood vessel segmentation. A summary of the performance of the described methods may be seen in table 3.6. Such methods are grouped according to the learning technique and are presented in chronological order.

The fact that an extremely high percentage of the presented approaches used both the DRIVE and STARE databases allows a more simple, intuitive and incisive analyse and comparison of the performance indicators. However, in some of the methods, there are still a lack of consensus on which performance indicators should be considered to analyse and compare the overall performance of the algorithms.

Different methods based on different learning techniques have been proposed over time. The methods using unsupervised learning and based on morphological processing and vessel tracking/tracing were the first to be proposed, studied and implemented. The results of these implementations shown that the combination of different approaches delivers the best results considering the sensitivity and accuracy.

In recent years, the use of supervised leaning has grown exponentially. The implementation of machine learning based methodologies have shown an important improvement on specificity and accuracy, when compared to the unsupervised learning techniques.

Although machine learning methodologies have presented interesting results (as is example M.M Fraz *et al.* [3], Marin *et al.* [28] and Liskowski and Krawiec [51]) the application of these are still conditioned to the existence of a ground truth. Nonetheless, when labelled data is available, as is the case of retinal fundus images, deep learning has been further improving the performance of computers in many daily tasks. In this particular task, deep learning has also outperformed other methods, as shown in the work of Liskowski and Krawiec [51].

Table 3.6: Performance analysis of the methods presented in Chapter 3.

Performance indicators analysis								
Learning Technique	**Methodology**	**Algorithm**	**Year**	**Database**	**Sensitivity**	**Specificity**	**Accuracy**	**AUC**
2º Human Observer	-	-	-	DRIVE	0.7763	0.9723	0.9470	-
				STARE	0.8951	**0.9384**	0.9348	-
Supervised Learning	Traditional Machine Learning	Staal *et al.* [40]	2004	DRIVE	-	-	0.9442	0.9520
				STARE	-	-	0.9516	0.9614
		Soares *et al.* [29]	2006	DRIVE	-	-	0.9466	0.9614
				STARE	-	-	0.9480	0.9671
		Ricci *et al.* [41]	2007	DRIVE	-	-	0.9563	0.9558
				STARE	-	-	0.9584	0.9671
		Marin *et al.* [42]	2011	DRIVE	0.7067	0.9801	0.9452	0.9588
				STARE	0.6944	**0.9819**	0.9526	0.9769
		M. M. Fraz *et al.* [43]	2012	DRIVE	0.7406	**0.9807**	0.9480	0.9747
				STARE	0.7548	0.9763	0.9534	0.9768
				CHASE_B1	0.7221	**0.9711**	**0.9469**	0.9712
	Deep Learning	Wang *et al.* [44]	2014	DRIVE	0.8173	0.9733	**0.9767**	0.9475
				STARE	0.8104	0.9791	**0.9813**	0.9751
		Melinscak *et al.* [48]	2015	DRIVE	-	-	0.9466	0.9749
				STARE	-	-	-	-
		Fu *et al.* [49]	2016	DRIVE	0.7294	-	0.9470	-
				STARE	0.7140	-	0.9545	-
		Liskowski and Krawiec [51]	2016	DRIVE	0.8149	0.9749	0.9530	**0.9788**
				STARE	0.9075	0.9771	0.9700	**0.9928**
Unsupervised Learning	Unsupervised	Salem *et al.* [18]	2007	DRIVE	0.8215	0.9750	-	-
				STARE	-	-	-	-
		Kande *et al.* [19]	2009	DRIVE	-	-	0.8911	0.9518
				STARE	-	-	0.8976	0.9298
		Villalobos-Castaldi *et al.* [20]	2010	DRIVE	**0.9648**	0.94803	0.9759	-
				STARE	-	-	-	-
	Match Filtering	Hoover *et al.* [22]	2000	DRIVE	-	-	-	-
				STARE	0.6751	0.9567	0.9267	-
		Zhang *et al.* [23]	2010	DRIVE	0.7120	0.9724	0.9382	-
				STARE	0.7177	0.9753	0.9484	-
	Morphological Processing	Mendonça *et al.* [27]	2006	DRIVE	0.7344	0.9764	0.9452	-
				STARE	0.6996	0.9730	0.9440	-
		M.M Fraz *et al.* [35]	2010	DRIVE	0.7152	0.9769	0.9430	-
				STARE	0.7311	0.9680	0.9442	-
	Vessel Tracking	Delibasis *et al.* [34]	2010	DRIVE	0.7288	0.9505	0.9311	-
				STARE	-	-	-	-
	Multi-Scale approach	Martinez-Perez *et al.* [36]	2007	DRIVE	0.7246	0.9655	0.9344	-
				STARE	0.7506	0.9569	0.9240	-

Chapter 4

Methodology

Multiple approaches based on deep learning for the segmentation of retinal blood vessels have been implemented. Regarding those presented on chapter 3, there are two different methodologies that use CNN's in distinct manners in order to segment the vessels. These are the cornerstones of the work presented in this chapter.

The first methodology, used by Liskowski and Krawiec [51], Melinscak *et al.* [48], and Fu *et al.* [49] is an element-wise classification where the CNN is trained patch by patch and the output is the probability of the center pixel (or a block of center pixels in the case of structured prediction) being in a certain class (in this case a two class classification, as each pixel may be vessel or non vessel). These approaches use a large number M of small square $N \times N$ patches from raw RGB or pre-processed training set of images as input of the neural network. The network is trained and validated with said data and then tested with patches from the test dataset.

Other methodology of work is proposed by Wang et al. [44], where the authors use CNN as deep feature extractor. The main process is similar to the one previously presented, as it starts with a large number M of small square $N \times N$ patches from a pre-processed training dataset of images that are fed as input to the neural network. In contrast to the other methodology, the activations of a specific group of layers are used to train different machine learning classifiers (the authors use the Random forest classifier). The test images undergo the same process and the features extracted are fed into the classifier. To finalize the process an ensemble mechanism is used, from which the final classification of is obtained.

Regarding the first scenario, three different network architectures were implemented, in order to analyze how the architecture of the network as well as different parameters can influence the segmentation result. In what concerns the second framework, the approach proposed by Wang *et al.* [44] was implemented, and more complex ensemble mechanisms were tested. In addition, a Support Vector Machine classifier (SVM) in combination with feature selection algorithms was also implemented.

In the following section, the deviated solutions are described, starting with the data pre-processing mechanism which was used. Then, a detailed description of the implemented networks is provided. Lastly, the experiments regarding the work of Wang *et al.* [44], more precisely on

feature selection, classification, and ensemble strategies are detailed.

4.1 Data preparation

The input data have a critical role in every machine learning algorithm, even in the deep learning based approaches. In theory, the high capacity of these systems should allow to equally learn from raw data, nonetheless the training process seems to benefit from an adequate pre-processing [51]. In this section, the approach that has been used to generate pre-processed patches of retinal images is presented.

4.1.1 Image Pre processing

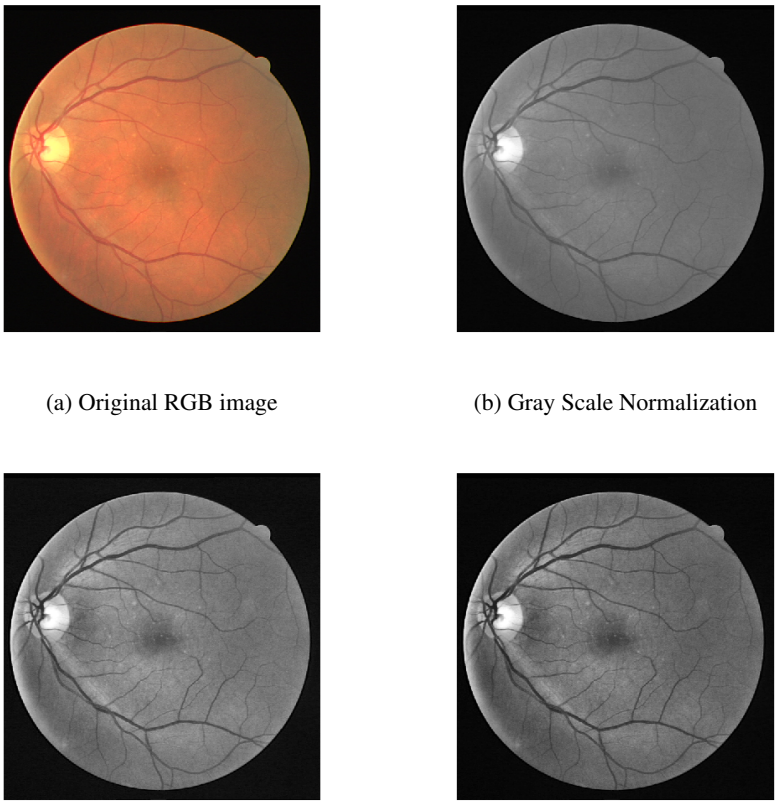
Several approaches may be employed to pre-process medical images. For instance, some works use the three color channels [51] and other rely only on the green channel intensity information [48], as it is the channel showing higher contrast between vessels and the background. The red and blue channels present low contrast and random variation of brightness [29].

The image pre processing technique used on this study starts with the information extracted from the green channel of the original RGB image (Figure 4.1a), then a Gray-scale normalization was applied due to the fact that as a result of acquisition process, very often retinal *fundus* images are non-uniformly illuminated and exhibit local luminosity and contrast variability (Figure 4.1b) [53].

An important aspect when considering the segmentation of the retinal vessel is the detection of the smaller vessels such as terminal vessels and small ramifications of the vascular tree, which frequently show low contrast to the background. To enhance their contrast, a contrast limited adaptive histogram equalization operating in 8x8 square tiles was performed [54]. This approach approximates the histogram of the intensities inside each square tile to a given histogram. Even if it does not largely improve the contrast of big structures, it makes the more subtle features in the image more pronounced (Figure 4.1c). To further improve the delineation of these smaller vessels, a gamma correction with a value of 1.2 was performed (Figure 4.1d). In Figure 4.1, the pre-processed image may be seen along the steps of the described pipeline. The presented pre processing approach was applied to both the training and testing datasets of the DRIVE and STARE databases.

4.1.2 Division into patches, data normalization and standardization

The extraction of small patches from retinal images allows the use of deep learning strategies to segment the vasculature, as the public databases usually do not contain more than few dozens of images. Afterwards, it is possible to frame the problem of vessel segmentation in two ways: classification of each patch, where the network outputs the probability of the center pixel belonging to a vessel [51]; and segmentation of each patch, where the network outputs a map of such probabilities, one for each pixel of the patch.



(a) Original RGB image

(b) Gray Scale Normalization

(c) Contrast Limited Adaptive Histogram Equalization

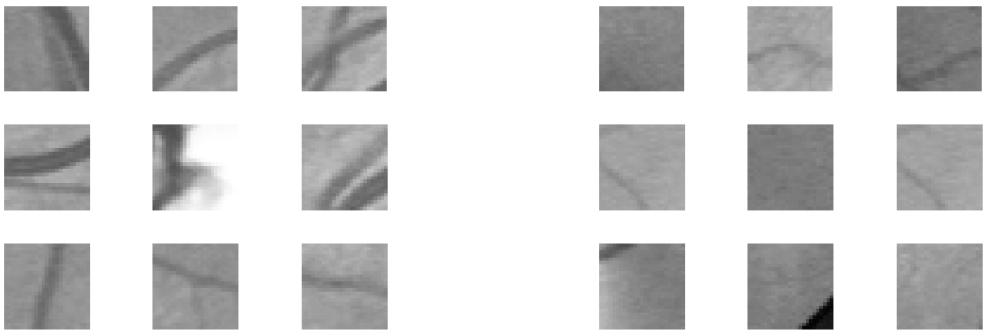
(d) Gamma Correction

Figure 4.1: Overall image pre-processing stages.

Regarding the sampling of patches from the images, the considered experiments varied between randomly sampling from the field of view; extracting a balanced set of patches, with regard to the label of the center pixel; and extracting a set of patches according to some arbitrary ratio $r \in [0, 1]$ of positive and negative samples. Examples of positive and negative patches, according to the view of a classification based approach, are presented in Figure 4.2 . The patch size, number, and strategy of sampling will be detailed with the description of each implementation.

In this thesis, only square patches are considered, as the majority of the literature suggests, in order to achieve a better coherence between the data and the network. In the prediction phase, in a patch classification setting, a patch has to be extracted for each pixel inside the field of view. Concerning the segmentation case, the computational cost is not as high, since less patches have to be extracted in order to obtain the final prediction image.

Data normalization and standardization The performance of machine learning based algorithms is heavily affected by the data used during training and evaluation processes. The use of methods to process data is standard to all machine learning approaches. Although the pre-



(a) (b)

Figure 4.2: Example of a positive set of patches (a) and a negative set of patches (b).

processing of data is a standard, the methods used can vary due to the type of data, the size and the algorithm used.

Using the studies of LeCun *et al.* [55] and Deng Li [56] on the performance and improvement of propagation and gradient decent based algorithms, two data transformations were implemented to boost the training performance: data normalization and data standardization.

Normalization is as simple as taking the ratios (reciprocal normalization), computing the differences (range normalization), and multiplicative normalization or Z-axis normalization. Normalization ensures that data is roughly uniformly distributed between the network inputs and the outputs [57]. In the presented case, the simple transformation given by the equation 4.1, was considered, yielding a scaling of the data into the range of [0,1].

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4.1}$$

Standardization is the most commonly used technique 4.2, transforming the data such that its arithmetic mean is zero and the standard deviation is one. However, both mean and standard deviation are sensitive to outliers, and this technique does not guarantee a common numerical range for the normalized scores [58].

$$x' = \frac{x - \bar{x}}{\sigma} \tag{4.2}$$

The data transformation used in combination with the type of activation functions applied are critical to the training performance and to the final results. The most used activation function in the hidden layers of a CNN is ReLu function, more specific the standard ReLU function which is given by the equation $f(x) = x^+ = max(0,x)$. This activation function allows to greatly accelerate the convergence of stochastic gradient descent compared to the sigmoid/tanh functions, it prevents the saturation and is very efficient in terms of computation [59].

Regarding the output layer, as the final goal is a binary classification (whether a pixel is vessel or non vessel) on these cases the most commonly used activation function used are Softmax and Sigmoid. The softmax function 4.3 calculates the probabilities distribution of the event over 'n'

different events. In general way of saying, this function will calculate the probabilities of each target class over all possible target classes. The main advantage of using Softmax is the output probabilities range. The range will be 0 to 1, and the sum of all the probabilities will be equal to 1. The sigmoid function 4.4 has a simpler mode of operation, as it takes any range real number and returns the output value which falls in the range of 0 to 1.

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{4.3}$$

$$f(x) = \frac{1}{1+e^{-x}} \tag{4.4}$$

The idea shared by multiple authors [55] [60] is that the softmax activation is used for multiclass classification while the sigmoid function is used on most cases of a binary classification. During the work, different combinations of data processing and activation functions were tested. The best performance was attained by conducting data normalization and using the sigmoid activation function in the output layer.

4.1.2.1 Data augmentation

One thing to take into consideration during the training of a CNN is the fact that overfitting may occur due to the relatively small number of samples used as input comparing to the number of model parameters [61]. A strategy used to surpass this is to apply randomized transformations to the existing training data in order to virtually generate a newer and larger set of inputs. In this thesis, three different types of transformations were randomly combined in order to produce ten new images from each original one (some examples are shown in Figure 4.3), priorly to the extraction of patches. The considered transformations are a combination of:

- Scaling by a factor between 0.7 and 1.2.
- Horizontal or vertical flip.
- Random Rotation between [-90,90].

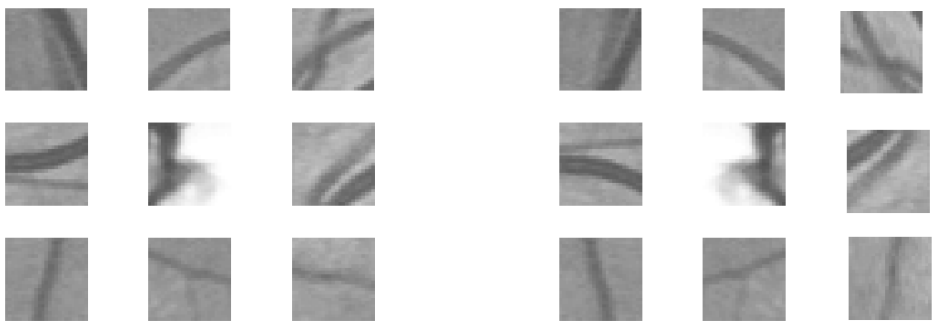


Figure 4.3: Original patches (left) and respective data augmentation transformation results (right).

4.2 Convolutional neural networks for retinal vessel segmentation

Convolutional neural Networks for semantic segmentation are being used in a large spectrum of research fields, in particular on the field of segmentation and classification of medical images. In this particular area of study, the use of Convolutional neural networks for a pixel-wise classification w has shown a solid set of interesting results in the detection and classification of a variety of anatomic structures such as tumors [62], multiple vessels [63], and brain lesions [64].

In this thesis, we explore and compare different network designs to tackle the problem of detecting retinal vessels in small patches: an U-net to achieve patch segmentation based on Ronneberger *et al.* [65] and in a MIT[1] implementation (section 4.2.2); the works of Liskowski and Krawiec [51] and Melinscak [48] for patch classification (section 4.2.1); and the use of deep features in random forests [44] and support vector machines (section 4.3).

4.2.1 Patch Classification approach

In a patch classification approach, the idea is to apply a CNN to a set of patches extracted from the image, and in each of them to predict the class label of the center pixel. Networks such as LeNet [55] and AlexNet [66] are examples of this design, where fixed size inputs produce non-spatial outputs. In figure 4.4, a typical architecture for classification is shown. It is common to have alternated steps of convolution and subsampling operations in order to capture increasingly higher level features which are appropriate for final decision.

The feature maps are subjected to one or more fully connected layers, from which the output is extracted. The proposed designs usually differ in terms of the number of convolutional, pooling and fully connected layers, the number and size of the kernels, and the inclusion of dropout and batch normalization layers.

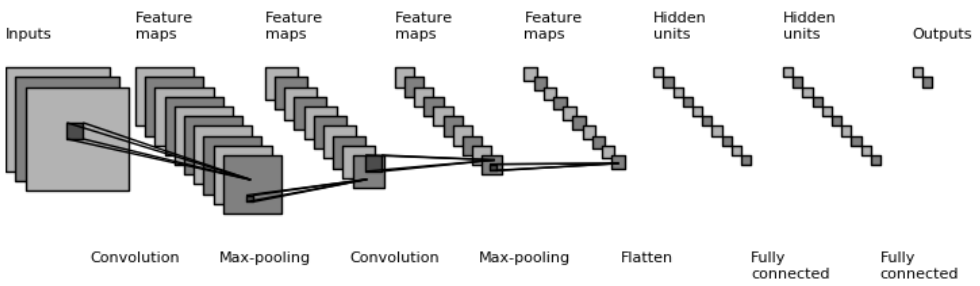


Figure 4.4: Example of the architecture of a classic Neural Network [55] .

In this thesis, two patch classification approaches are tested: the first one is based on the work of [48] and is used as baseline of work for this type of classification, since the architecture of the network and overall process is the most commonly used in this task. The second is based on the

[1] Source: https://github.com/orobix/retina-unet

approach presented by Liskowski and Krawiec [51], which is used to assess how the information of neighborhood pixels can be applied to improve the classification task (structure prediction).

4.2.1.1 Melinscak approach

Architecture The architecture starts with an input layer followed by a convolutional layer with 65 filters. The dimension of the employed kernels was 5×5 in order to differentiate between the vessel and the background by extracting small and local features. The use of a small stride value ensures that maximum information is retrieved.

Following this, a max pooling layer with a small 2x2 filter is applied, such that most promising features are carried over to the next convolutional layer. This process is repeated with a reduction in the number of maps and in the size of filters used in the convolutional layers (from a 5×5 to 3×3 and 2×2 in the final layer). The pooling layers and the stride value remain unchanged. This design was made by considering that neighboring pixels are strongly correlated (specially in the final layers), so it makes sense to reduce the size of the output by the filter response. The further apart two pixels are from each other, the less correlated. Therefore, a big stride in the pooling layer leads to high information loss [57].

The final layers of the network are two fully connected layers, the first one with 100 neurons, and the last one with two neurons from which the probability of each pixel of belonging to the background or the vessel structure is given. Regarding the activation functions used, as mentioned on 4.1.2, a ReLu activation function was used in the hidden layers, and a sigmoid activation function was considered in the last dense layer, as the classification problem here is binary for each pixel, and as mentioned before, the sigmoid leads to a simpler and faster process. The final architecture of the network used in this approach in presented on table 4.1.

Dropout and Batch Normalization layers As referred before, the pre-processing of the data used in the training process is important for the performance of a CNN. Beside this, other strategies regarding the network architecture have been tested to achieve the same goal. Often, improvement is achieved by using Batch normalization and Dropout layers [56].

Batch normalization is a technique used to provide any layer in a CNN with inputs that have zero mean/unit variance. The layers first normalize the activations of each channel by subtracting the mini-batch mean and dividing by the mini-batch standard deviation, the equations of this process are presented on Figure 4.5. Then, the layer shifts the input by a an offset β and scales it by a scale factor γ. Both of these variables are learnable parameters that are updated during the network training.

The Batch normalization layers are used between the linear and non-linear layers, generally between the convolutional layer and ReLu. By reducing the non-linearities between layers and updating the γ and β parameters, this type of layers helps to reduce the sensitivity of the network to initialization, speed up the network training and prevent overfit [66].

One of the fist approach that used Dropout layers was presented by Srivastava *et al* [67]. This is a regularization technique where randomly selected neurons are ignored during training.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Figure 4.5: Equations used during the application of batch normalization layer [66] .

The contribution of the ignored neurons to the activation of downstream neurons is temporarily removed on the forward pass and any weight updates are not applied to the neuron on the backward pass [66] [56] . This process is illustrated on Figure 4.6.

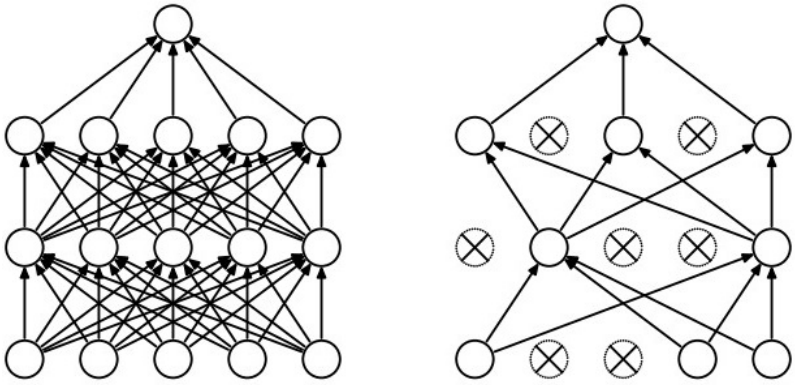


Figure 4.6: Illustration of the dropout process. On the left is an example of the standard procedure and on the right is the result of the Dropout [67].

As a neural network learns, neuron weights settle into their context within the network. Weights of neurons are tuned for specific features providing some specialization. Neighboring neurons end relying on this specialization, which if taken too far can result in a fragile model too specialized to the training data.

During the work, two different approaches were tested: Batch normalization layers were employed between convolutional layers and the ReLU activations and dropout was used on fully connected layers. The results have shown that the best performances were obtained when using the dropout technique, with the decrease of the training time by 5s per epoch and the increase of the accuracy. The ratio of neuron dropped during this process was 20%, the same in all of the

cases and the final architecture used is presented on table 4.1.

Table 4.1: Architecture of the implemented Convolutional Neural Network.

Layer	Type	Maps and Neurons	Kernel size	Stride	Padding
0	Input	1 Map of 65x65 Neurons	-	-	-
1	Convolutional	48 Maps of 60x60 Neurons	5x5	1	0
2	Max-Pooling	48 Maps of 30x30 Neurons	2x2	2	0
3	Convolutional	48 Maps of 28x28 Neurons	3x3	1	0
4	Max-Pooling	48 Maps of 14x14 Neurons	2x2	2	0
5	Convolutional	48 Maps of 12x12 Neurons	2x2	1	0
6	Max-Pooling	48 Maps of 6x6 Neurons	2x2	2	0
7	Fully connected	100 Neurons	1x1	-	-
8	Dropout	20% rate	-	-	-
9	Fully connected	2 Neurons	1x1	-	-

Training In this implementation the ADAM optimizer was used, an algorithm for first-order gradient-based optimization of stochastic objective functions based on adaptive estimates of lower-order moments. This was chosen since it is computationally efficient with little memory requirements and is invariant to diagonal rescaling of the gradients. The loss function used was binary cross entropy in virtue of the binary classification, and 150 epochs were used to train the network.

The starting learning rate used was 1e-3, this value was decremented by a factor of 0.01 each time the validation accuracy or loss kept the same value for more than a period of 5 epochs to a limit value of 1e-6.

4.2.1.2 Liskwoski - Structure Prediction

Among the works analyzed in the state-of-the-art (Chapter 3), the most promising results were achieved by Liskowski and Krawiec [51]. The authors assessed the impact of preprocessing and network design in the performance. Their best results were achieved by classifying not only the center pixel of the patch, but instead a whole square block of pixels in the center, a mechanism which they refer to as structure prediction.

The motivation lies in the high correlation that usually is seen at nearby pixels of medical images, such that spatial relations between these pixels influence their odds of representing particular structures. Thus, classifying a set of neighboring pixels at the same time seems a natural approach to exploit such fact.

Architecture The architectures used is this implementation were similar to the ones presented on the previous section. Different types of pre-processing transformations (Global contrast normalization, ZCA withing and Data augmentation) are used to train two similar architectures in order to discover which combination presented the best metrics results. The combination (Data

and architecture) that provided the best results are then used as base for the implementation of the structure prediction.

Since the work presented by Liskowski and Krawiec [51] was the one that yielded the best results of all the deep learning approaches that were analyzed on chapter 3, the methodology used in this work was very similar to the one presented by the authors. The first architecture is composed by two convolutional layers each with 64 filters followed by a max-pooling layer. This configuration was repeated with a change on the number of filters used on the convulotional layers (an increment to 128). The final layers are fully connected with 512 neurons with exception of the last one, were only two exist for classification purposes.

Considering the second architecture, it is the same as the first with the exception that no max pooling layers were used, as the size of patches are already very small (square patches of 25x25 pixels) and as shown in [68] networks with convolution layers only might perform better when applied to small images. The rest of the architecture is the same as described before. The final architecture is presented on table 4.2.

Table 4.2: Architecture of the implemented Convolutional Neural Network based on the approach proposed by Liskowski.

Layer	**Type**	**Maps and Neurons**	**Kernel size**	**Stride**	**Padding**
0	Input	1 Map of 25x25 Neurons	-	-	-
1	Convolutional	64 Maps of 23x23 Neurons	3x3	1	0
2	Convolutional	128 Maps of 21x21 Neurons	3x3	1	0
3	Convolutional	128 Maps of 17x17 Neurons	3x3	1	0
4	Fully connected	100 Neurons	1x1	-	-
5	Fully connected	100 Neurons	1x1	-	-
6	Fully connected	*9 Neurons*	1x1	-	-

Regarding the data, in both methodologies, either the raw RGB images or their preprocessed versions obtained according to the procedure described section 4.1.1 were used. The number of positive and negative patches used varied from a 50/50 ratio, 20/80 ratio and random samples.

Structure Prediction Following this approach, the segmentation task poses as a multi label inference problem on a set of binary predictions subject to a joint loss. Instead of considering only the central pixel when retrieving the information from the ground truth, now the information of all the pixels contained in a small square window of dimension s centered on the central pixels is regarded.

The architecture of the network used in the structure prediction was the one that yielded the best results of those presented before. Considering the new multi label classification of the structure prediction technique, the final two neuron fully connected layer of the network was replaced by a dense layer with s^2 neurons from where the final classifications are extracted, an example of this is presented on the Figure 4.7.

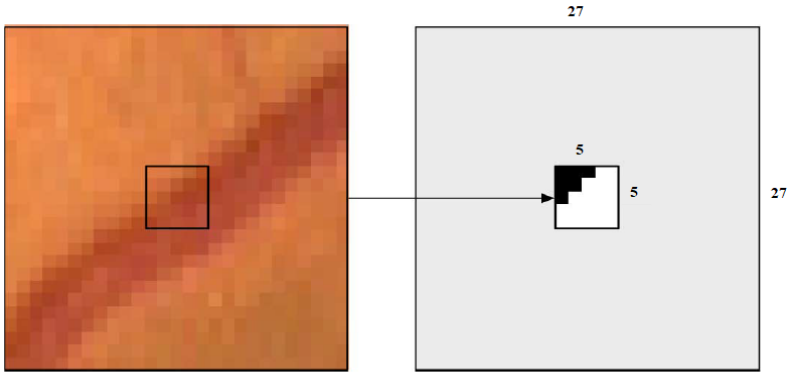


Figure 4.7: A training example for the SP approach: the 27x27 patch (left) with a s x s = 5 x 5 and the corresponding desired output (right) [51].

Although the authors tried three different values for s (s= 3,5,9), during this implementation the size of the window used was 3x3. This value was chosen in virtue of the interesting results obtained by the authors and taking into consideration the complexity of the process used to extract the information of border pixels. This value of s did not require a significant alteration in the extraction process and minimized the loss of information.

Training Similarly to those presented before, in these implementations ReLU activation functions were used in the hidden layers and for the final layer was selected the Sigmoid function. Training was carried out by stochastic gradient descent with a starting learning rate of 1e-3, this value was decremented by a factor of 0.01 each time the validation accuracy or loss kept the same value for more than a period of 5 epochs to a limit value of 1e-6.

4.2.2 Patch Segmentation approach

In this approach, the CNN predicts the probability of each pixel in the patch to belong to a vessel. The U-net [65] is an encoder-decoder type network architecture for image segmentation. Its name comes from its unique shape, where the feature maps of decreasing resolution in the downsampling step are fed into the convolutional layers in the upsampling step (skip-layer connections). The contracting path captures contextual information by successively reducing the resolution of feature maps, whereas the connection path combines feature maps from the contracting and expanding paths to fuse local information with accurate localization [65].

The contracting path follows the typical architecture of a convolutional network. It consists of the repeated application of two 3x3 convolutions (unpadded convolutions), each followed by a ReLU and a 2x2 max pooling operation with stride 2 for downsampling [65]. Every step in the expansive path consists of an up sampling of the feature map followed by a 2x2 convolution (“up-convolution”) that halves the number of feature channels, a concatenation with the correspondingly

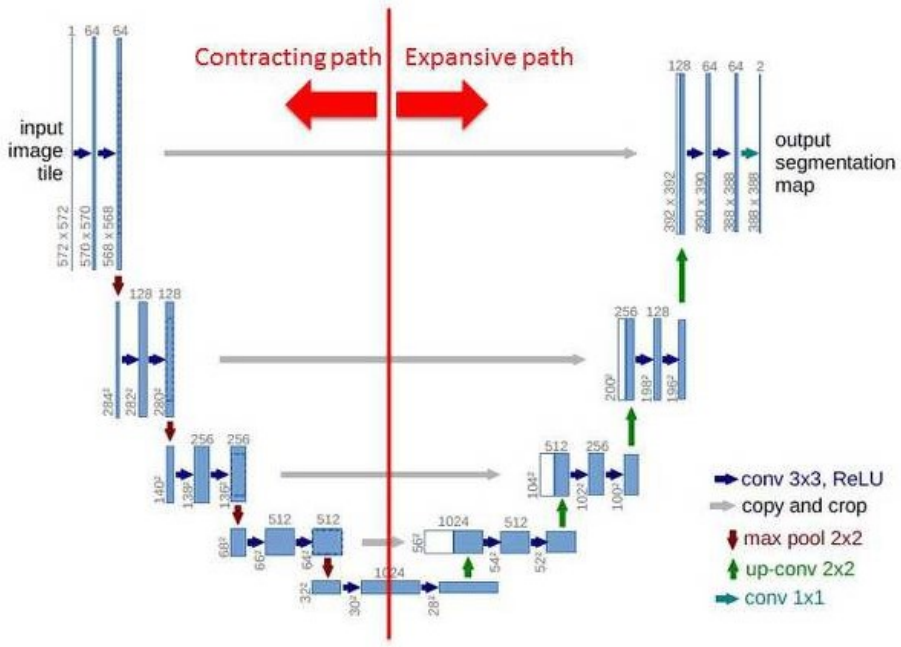


Figure 4.8: Example with the schematics and description of a U-net architecture with the identification of the contracting and expanding paths [65].

cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU [65].

This implementation was based in the previously work provided under MIT License[2]. Following the structure presented on Figure 4.8, this implementation additionally uses dropout layers between the two convolutional layers at each resolution level. As suggested by [69] one might think that since the convolutional layers do not have a lot of parameters, overfitting is not a problem and therefore dropout would not have much effect. However, the use of dropout in the lower layers helps because it provides noisy inputs for the higher layers which prevents them from overfitting in particular cases such as this one.

In table 4.3 the backbone of network used is presented. The final layer is a convolutional layer from which the final classification results are obtained. The softmax function was used in the last layer.

To train this implementation, were used pre-processed images that underwent the process explained on section 4.1.1, from each image were extracted 9500 square patches of 48×48 dimension randomly extracted from the FOV region. Part of the training dataset was reserved for validation, more precisely 10% of the extracted patches were set aside for validation purposes.

The training procedure ran for 150 epochs, using a batch size of 32 patches. A stochastic gradient descent optimizer was applied with a learning rate of 0.01, a decay of 1e-6, and a momentum

[2]Source: https://github.com/orobix/retina-unet

Table 4.3: Structure of the implemented U-Net.

Path	Layer	Type	Maps size	Kernel size
Contracting	1	Convolutional	32	3x3
	2	Dropout	-	-
	3	Convolutional	32	3x3
	4	Max Pooling	-	2x2
	5	Convolutional	64	3x3
	6	Dropout	-	-
	7	Convolutional	64	3x3
	8	Max Pooling	-	2x2
	9	Convolutional	128	3x3
	10	Dropout	-	-
	11	Convolutional	128	3x3
Expansive	12	UpSampling	-	2x2
	13	Convolutional	64	3x3
	14	Dropout	-	-
	15	Convolutional	64	3x3
	16	UpSampling	-	2x2
	17	Convolutional	32	3x3
	18	Dropout	-	-
	19	Convolutional	32	3x3
	20	Convolutional	2	1x1
	21	Activation Softmax	-	-

of 0.3. Since the softmax activation function was used, the categorical crossentropy loss function was considered.

4.3 Convolutional neural networks for deep features extraction

The methodologies presented so far use deep neural networks in their complete extension to make predictions. However, another possibility is to use a CNN to learn representative feature maps and then use such features as the input to other learning algorithms, such as Random Forests (RF) and support vector machine (SVM) [55].

Initially, methodologies employing this strategy follow the same training procedure, in order to learn CNN's weights. The difference comes after this first training, as the deep learned features will be used to train another learning algorithm. In this thesis, we explore the work of [44], where a modified version of LeNet-5 [55] is trained via a patch-based classification approach, and afterwards an ensemble of Random Forests uses deep features to make predictions. Each RF is trained on the deep features of a single abstraction level, as shown in Figure 4.9. The authors considered a winner-takes-it-all ensemble when making predictions. Besides testing this

methodology, other variations are explored in this thesis, namely other ensemble strategies, and the use of a Support Vector Machine (SVM) as learning algorithm.

4.3.1 Feature extraction

Considering that the training time tends to scale with the amount and size of training data, and the complexity of the network [44], in this implementation the process of feature extraction is carried out by a modified version of the classic LeNet-5 architecture. The used version is the modification proposed in [55], where 4x4 convolution filters and subsampling by factor 2 are used. Dropout layers with probability of 20% were also considered. This design is presented in Table 4.4.

The training procedure was conducted using 10500 square patches of size 25×25 from each training image, leading to a total of 210000 patches. The patch extraction occurred according to the process described in section 4.1.2. The decision to use this patch size was based on the constraints mentioned before regarding training time and considerations about the average size of the lesions that exist on some images of the datasets [44].

Table 4.4: Structure of the LeNet used for feature extraction.

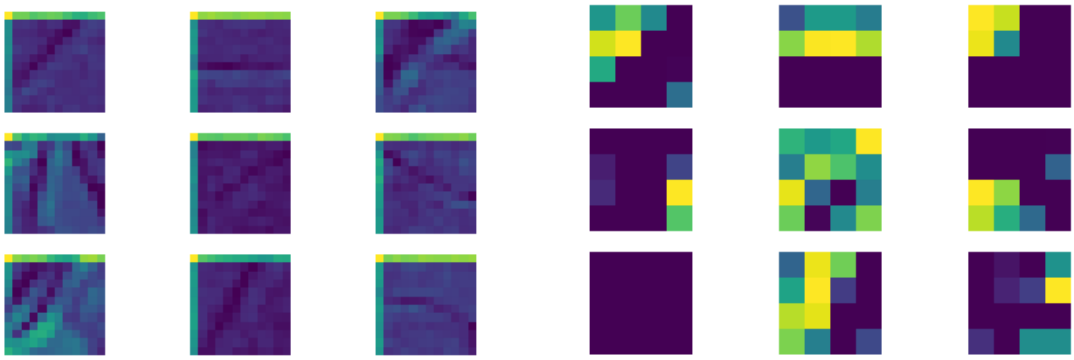
Layer	Type	Maps and Neurons	Kernel Size	Stride	Padding	Reference
0	Input	1 Map of 25 x 25 neurons		-	-	
1	Convolutional	12 Maps of 22 x 22 neurons	4 x 4	1	1	C1
2	Max-pooling	12 Maps of 11 x 11 neurons	2 x 2	2	0	MP01
3	Convolutional	12 Maps of 8 x 8 neurons	4 x 4	1	1	C2
4	Max-pooling	12 Maps of 4 x 4 neurons	2 x 2	2	0	MP02
5	Convolutional	100 Neurons	4 x 4	1	1	C2
6	Output	1 neurons		-	-	

Using Table 4.4 as reference, in [44], the authors used the feature maps from MP01, MP02, and C3 as inputs to the Random Forest classifiers. These layers were selected considering the balance between the obtained information and the number of extracted features. As a result, the first pooling layer (MP01) extracts elementary features with high resolution, whereas the maps coming from the other layers (MP02 and C3) capture high-order statistics in lower resolutions. Example maps extracted from these layers are presented in Figure 4.9.

4.3.2 Classifiers

In contrast with the approaches described in section 4.2, here traditional machine learning algorithms are used to infer the final label of each pixel. In this work, two different classifiers are tested, namely Random Forest and SVM, which have been previously used in similar classification tasks [44].

These learning algorithms were chosen considering the findings in [55], where they are reported as the most commonly used and the ones achieving overall better performances in several binary classification tasks.



(a) (b)

Figure 4.9: Example of features extracted from (a) MP01 layer and from MP02 (b).

4.3.2.1 Random Forest

The Random Forest algorithm is one of the most vastly used algorithm for either regression or classifications tasks. Firstly proposed by Breiman [70] this algorithm consists of tree predictors, where each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest.

Each tree grows as follows: First, the training set used to grow each tree is a boot strap sample of the observations. Some observations are represented multiple times, while others are left out. The left-out observations are called "out-of-bag" and these out-of-bag samples can be used to calculate an unbiased error rate and variable importance, which eliminates the need for a test set or cross-validation. Following this, the best split at each node is selected from among a random subset of the predictor variable. In the final step of this process, each tree is grown to its large extent possible. Classification is given by a voting mechanism. The process is illustrated in Figure 4.10.

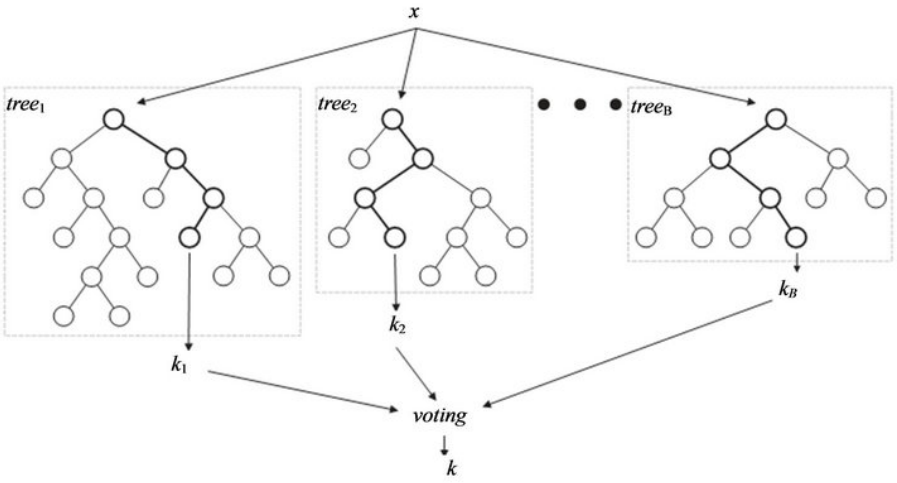


Figure 4.10: Illustration of how the random forest algorithm preforms a classification task [70].

Considering now the features extracted using the CNN, the number of features extracted from each patch vary depending on the layer where these were extracted. The max number of features extracted for one patch is 1840 (Layer MP01), resulting on a feature set with 210000×1840 dimensions. The large number of features extracted for each patch creates a large and complex dataset. One of the key aspects for the selection of the Random Forest algorithm as the implemented classifier is it robustness and the ability to work with large dataset without the need to perform feature selection, which allows to use all the features retrieved by the CNN [70].

As explained before, the features extracted from different layers provide different types of information, thus three Random Forest classifiers were trained each one using the data from a specific layer. The three Classifiers used shared the same hyper-parameters.

In general, to implement the Random Forest algorithm the number of hyper parameters necessary to take into consideration is relatively small and straightforward to understand. Those are used to increase the predictive power of the model or to make the model faster. Of those, the most important is the number of estimators (trees) used during the process [53]. In this implementation pruning was not used and gini impurity was considered instead of entropy, because it has a lower computational cost than the latter, which requires calculating the logarithmic function.

In this thesis, due to the characteristics and the size of the feature dataset, 100 estimators were used, this number emerged from the commitment between the performance and processing time. As confirmed during the implementation, a greater number of estimators would require a significant increase in the amount of time required for computation and the improvement of performance did not justify the effort, considering the limited time schedule and the resources available.

4.3.2.2 Support vector machine

Support Vector Machines are computational learning methods that transform a classification problem in such a way that allows the application of linear classification techniques to nonlinear data. SVMs belong to the class of Kernel Methods that are formed by a general purpose learning machine and a problem specific kernel function. Since linear machine learning methods can only classify data in a linear space with separable features, the kernel-function can be used to obtain the linear feature space by mapping the training data into a higher dimensional space where the data is linearly separable. These Kernel functions can be of many types, from polynomial to Gaussian (Radial Basis) or Sigmoid.

Having a certain training data set that comprises N input vectors $x_1 ... x_n$ with matching target values $t_1 ... t_n$, with t_n being either -1 or 1 the goal is to separate that set into several individual classes. The SVM approaches this problem through the concept of margin, i.e. the largest distance between the decision boundary and any of the samples. The margin's value is given by the perpendicular distance to the closest point from the data set t_1 so, by optimizing the parameters w and

b, the margin is maximized [71]. This optimization problem is represented by the equation 4.5.

$$argmax_{w,b}\left\{\frac{1}{\|w\|}min_n[t^n(w^T\phi(x_n)+b)]\right\} \tag{4.5}$$

In this thesis three SVM classifiers were used, all sharing the same hyper parameters, and each of them trained with the features retrieved from a different layer of the CNN (the same process that was used in the Random Forest case). In contrast with the Random Forests, the performance of these classifier is heavily influenced by the features that are considered. Taking into consideration the number of features, their values and characteristics the kernel used was the Radial basis function and the value of the margin was set to 1.

In consequence of the high number of patches extracted and the number of features of each patch, the initial tests with SVM were a very time-consuming process (approx. 5-8 days). To overcome this and improve the results of the classification, two feature selection mechanisms were employed: Univariate Selection and Recursive feature selection with cross-validation.

Feature Selection - Univariate Selection The downside of having a large dataset of features is that some of them are irrelevant and they can decrease the accuracy of many models, especially linear algorithms like linear and logistic regression. The use of feature selection can improve the accuracy, reduce overfitting and the time needed to train the classifier [72].

Univariate feature selection works by selecting the best features based on univariate statistical tests. This method examines each feature individually to determine the strength of the relationship of the feature with the response variable [72].

In this dissertation, this method was use to reduce the number of features to 75% and 90% of the size of the original feature dataset. The features were selected based on a chi^2 statistical test.

Feature Selection - Recursive feature selection with cross-validation Recursive feature elimination is based on the idea to repeatedly construct a model (for example a SVM or a regression model) and choose either the best or worst performing feature (for example based on coefficients), setting the feature aside and then repeating the process with the rest of the features. This process is applied until all features in the dataset are exhausted. Features are then ranked according to when they were eliminated. As such, it is a greedy optimization for finding the best performing subset of features.

The dataset is divided into N folds, as some of those are used to improve the model and the rest is used in the cross validation process.

In this implementation, this process was performed using only a two fold split due to the number of samples used during the process.

4.3.3 Ensemble

The result of each classifier is dependent on the features used during the training process: as features extracted from different layers provide different types of information and each classifier is

trained with information extracted from a single abstraction level, the final result of the classification is heavily restrained by the type of data used.

It is expected that features coming from different level abstractions are relevant for the classification process, such that ensembling the classifiers trained at different resolution levels may lead to improved outcomes.

Wang *et al.* [44], only considered the best performing classifier, in this thesis the contribution of each classifier is taken into consideration: the first ensemble mechanism is the selection of best results from the three classifier ("winner takes it all"), the second is the average of probability given by each classifier and the third ensemble technique is the selection of the minimum probability.

The final ensemble mechanism used in this thesis is based on a "fuzzy-like" system, where the weight of a classifier in the calculation of the final prediction is determined based on their accuracy.

Chapter 5

Results and Discussion

In this chapter, the results of the implementations described on chapter 4 are presented. Similar to what was done in the previous chapter, the results are presented accordingly to the classification approach used. Following the presentation of the results, a discussion and comparison to those presented on chapter 3 is provided. The details of each implementation are also presented.

5.1 Results of CNN based approaches for retinal vessel segmentation

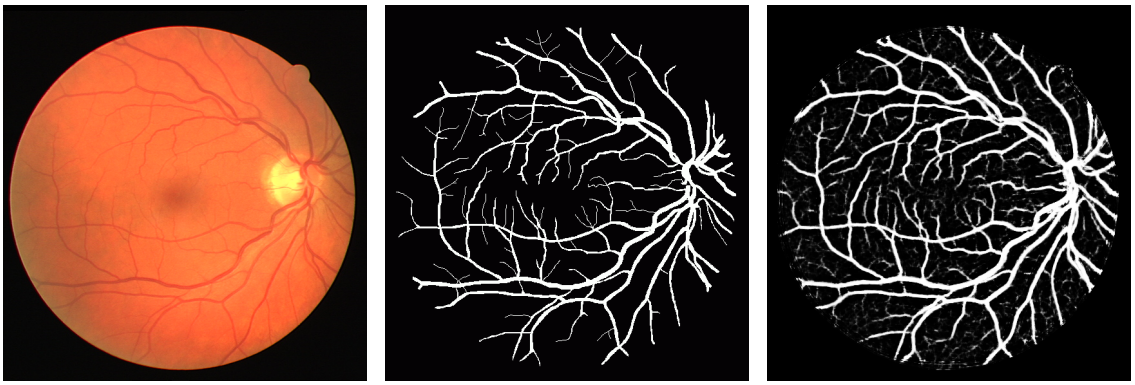
5.1.1 Patch Classification approach

5.1.1.1 Results of the Melinscak based approach

The first implementation, described in section 4.2.1.1, was trained using pre processed images from the DRIVE and STARE dataset. Regarding the procedure on DRIVE: the train set (20 images) was divided into a training set (15 images) and a validation set (5 images). To test the implementation was used the test set (20 images) .

In some of the work presented before (chapter 3) the training process using the STARE dataset was done resorting to the *leave-one-out* validation: the training and testing cycle is repeated 20 times, and in each iteration 19 of the 20 *fundus* images form the source of training examples, and the remaining image is the source of test patches. In this thesis, due to the limitation in terms of time and resources needed to evaluate all the implementations, this dataset (20 images) was divided into two: a training set (14 images for training and 1 for validation) and a test set (5 images).

The overall process of this implementation on the DRIVE dataset took approximately 12 hours, using a NVIDIA GeForce GT 650M graphics card. The results are presented on table A.1. This methodology lead to an average accuracy of 0.9101, a specificity value of 0.9142, a sensitivity of 0.8925 and precision of 0.6115. The average AUC value was 0.9604, where minimum AUC was 0.9450 and maximum 0.9710. In Figures 5.1a-5.1c, are presented the original image, the ground truth and the best accuracy result of this implementation.

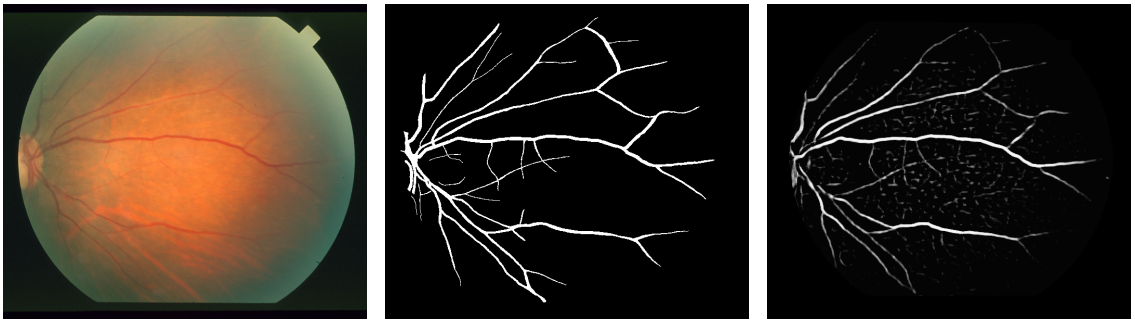


(a) (b) (c)

Figure 5.1: Original DRIVE image (a), its manual segmentation (b) and Best overall result of the Melinscak based approach on the DRIVE dataset (c).

Considering the STARE dataset, the overall process took considerably less time (aprox. 5 hours) due to the smaller size of the dataset. The same graphics card was used on this process. The results of this implementation are presented on table A.2. The average results obtained for accuracy, specificity, sensitivity and precision were: 0.9424, 0.9847, 0.5390 and 0.8345 respectively.

The AUC average value was 0.9375, where minimum AUC was 0.9150 and maximum was 0.9479. The best accuracy overall results were obtained on a image with a less complex vascular tree as is visible in figure 5.2c. The original image is presented in Figure 5.2a and the manual segmentation is presented in 5.2b.



(a) (b) (c)

Figure 5.2: Original STARE image (a), its manual segmentation (b) and Best overall result of the Melinscak based approach on the STRARE dataset (c).

5.1.1.2 Results of the Liskowski based approach

As described on 4.2.1.2 , the approach based on the work of Liskowski and Krawiec [51] was trained using different ratios of patches extracted from pre-precessed images: 20% of positive patches and 80% of negative patches (Liskowski SP), an equal percentage of positive and negative (liskowski SP Balanced) and a random percentage of each (liskowski SP Random). This approach

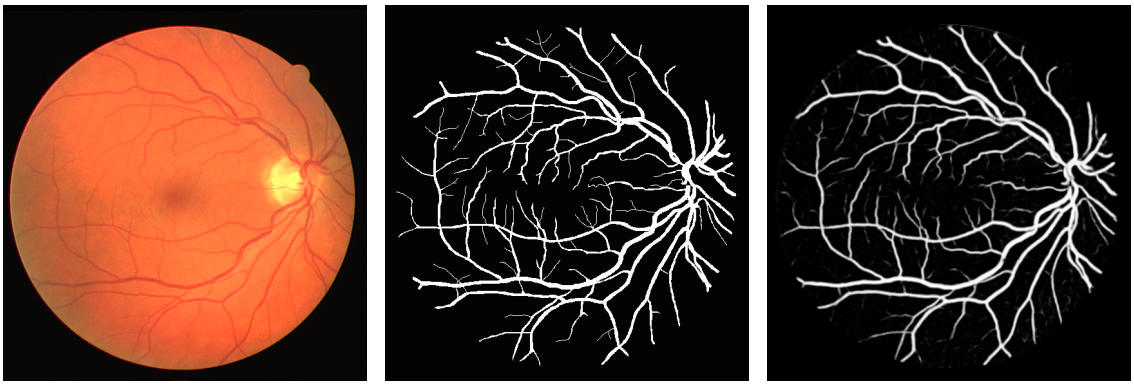
was also trained using patches extracted from the original RGB images with 20/80 ratio (liskowski SP RGB).

In these implementations the DRIVE and STARE datasets were used in the same way as described on 5.1.1.1.

Regarding the DRIVE dataset each implementation took on average 22 hours using a NVIDIA GeForce GT 650M graphics card. The detailed results are presented on tables A.3 to A.6. The average of the results are summarized and presented on table 5.1. Comparing the results presented on table 5.1, the best overall preforming implementations are the ones that used random sampling and 20/80 ratio. In Figure 5.3a is presented the original image, in Figure 5.3b the manual segmentation, and in Figure 5.3c the result of the Liskowski SP implementation. Regading the Liskowski SP random, in figures 5.4a, 5.4b and 5.4c are presented: the original image, the manual segmentation and the result, respectively.

Table 5.1: Average results of the Liskowski based implementations on the DRIVE dataset.

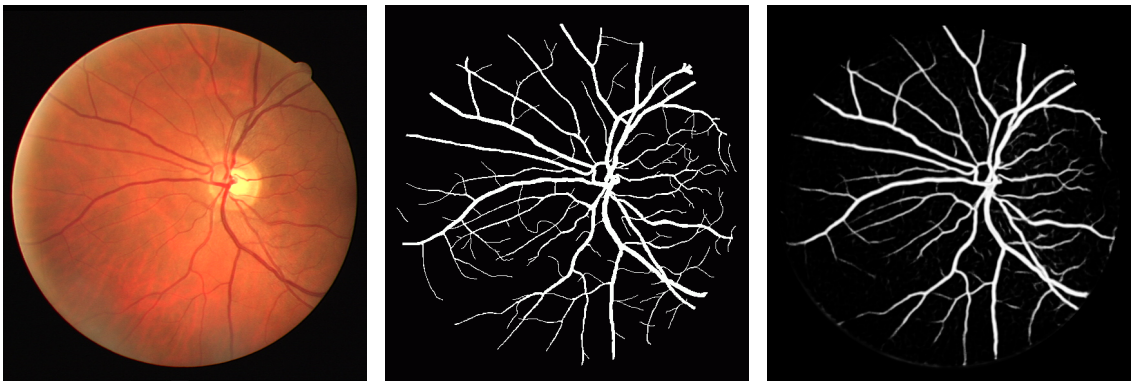
Algorithm	Dataset	Acc	SP	Sn	Pr	AUC
Liskowski SP (20/80)	DRIVE	**0.9434**	0.9515	**0.8400**	0.7233	**0.9677**
Liskowski SP Balanced (50/50)		0.9310	0.9466	0.7990	0.6901	0.9520
Liskowski SP Random		0.9415	**0.9751**	0.6973	**0.8143**	0.9489
Liskowski SP RGB (20/80)		0.9405	**0.9751**	0.6994	**0.8115**	0.9522



(a) (b) (c)

Figure 5.3: Original DRIVE image (a), its manual segmentation (b) and Best overall result of the Liskowski SP based approach on DRIVE dataset (c).

Considering now the implementations trained on the STARE dataset, the results of these are presented on tables A.7 to A.10. The average results of these are presented on table 5.2, from these, the best overall performance was yielded from the implementation that used a 20/80 ratio, equal to the DRIVE dataset. The implementation took on average 13 hours, on the same graphics card as mentioned before. The results of this implementation are presented in image 5.5a (original image), image 5.5b (the manual segmentation and image) and in 5.5c the result.



(a) (b) (c)

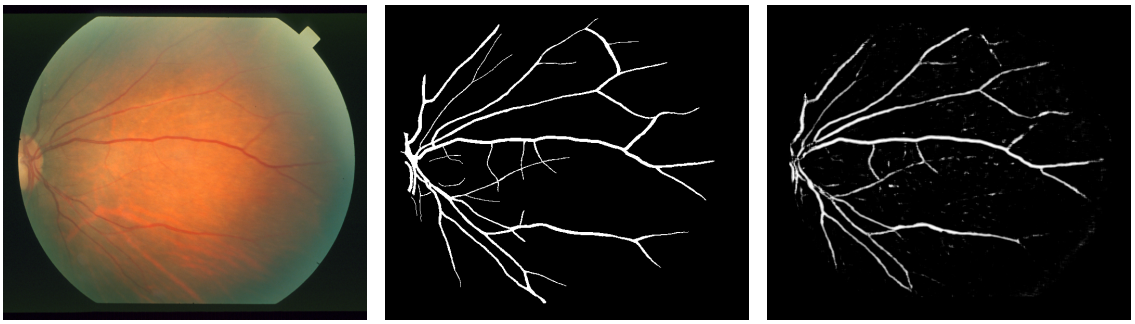
Figure 5.4: Original DRIVE image (a), its manual segmentation (b) and Best overall result of the Liskowski Random based approach on DRIVE dataset (c).

Table 5.2: Average results of the Liskowski based implementations on the STARE dataset.

Algorithm	**Dataset**	**Acc**	**SP**	**Sn**	**Pr**	**AUC**
Liskowski SP (20/80)	STARE	**0.9485**	**0.9656**	0.8207	0.7973	**0.9565**
Liskowski SP Balanced (50/50)		0.9391	0.9621	0.8103	**0.8014**	0.9549
Liskowski SP Random		0.9450	0.9644	0.8033	0.7926	0.9557
Liskowski SP RGB (20/80)		0.9427	0.9590	**0.8256**	0.7982	0.9555

Considering all the results of the patch classification approaches, the best performances are yielded from implementations based on the work of Liskowski and Krawiec [51]. As these take into consideration the surrounding information to classify the central pixel of each patch. The use of this information improved the segmentation of the smaller and terminal vessel as well as the delimitations of the larger vessels.

From this, the accuracy improved from an average of 0.9101 (Melinscak based approach) to 0.9434, and the value of AUC also had a minor increase. The best preforming implementations



(a) (b) (c)

Figure 5.5: Original DRIVE image (a), its manual segmentation (b) and Best overall result of the Liskowski Random based approach on DRIVE dataset (c).

were also the ones trained using a small number of positive patches or a random ratios. This is due to the fact that the probability of the central pixel of a patch being a vessel is lower than it is of being a background pixel. Using these ratios allows the network to be trained with more relevant data and prevents the number of false positive detections, and consequently better results. Nonetheless, there is also a decrease on the fine vessels that are segmented.

On both approaches, the best results from STARE dataset came from the same image: one with a simpler vascular tree and without thinner terminal vessels. This had impact on the calculation of the metrics, and is an indirect consequence of the limited number of test images in this dataset.

Comparing these approaches with the one in table 3.6, which are based on deep learning, the average results obtained are worse. This can be caused by multiple aspects and details not disclosed on the articles used as base for these implementations. When compared to the traditional machine learning approaches, the obtained results (accuracy and AUC) outperform the work presented by Marin *et al.* [42] and Stall *et al.* [40] on DRIVE dataset and AUC on STARE dataset.

5.1.2 Patch Segmentation approach

5.1.2.1 Results of the U-Net approach

Regarding now the patch segmentation approach, as mentioned before this was based on an MIT implementation[1]. To train the network, were used the DRIVE and STARE dataset in the same way as described on 5.1.1.1.

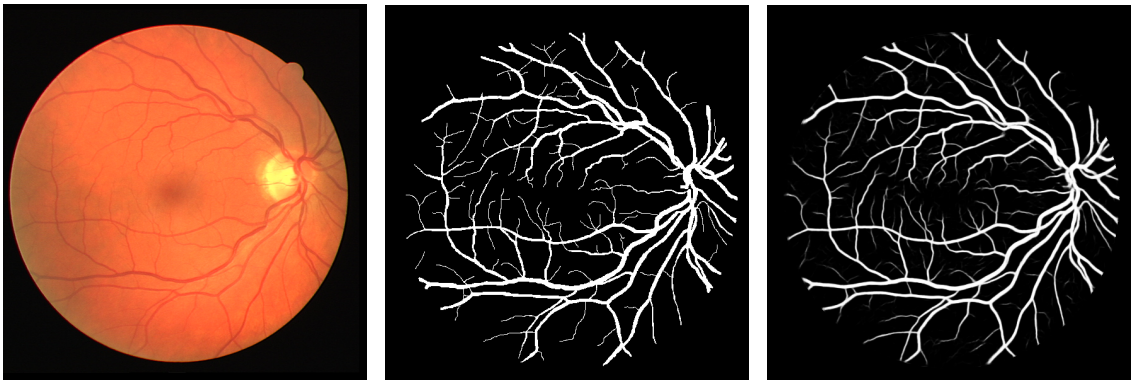
Using a NVIDIA GeForce GT 650M graphics card, the overall process took 23 hours when using the DRIVE dataset, and about 14 when using STARE. In contrast to the implementations presented on the previous chapter where the performance of the implementation was given for each image individually, in this case the performance of approach is given for the whole dataset for an overall analysis of the results. In table 5.3 are presented the results obtained for each dataset.

Table 5.3: Average results of the U-net implementation on DRIVE and STARE datasets.

Algorithm	Dataset	Acc	SP	Sn	Pr	AUC
U-Net	DRIVE	**0.9559**	**0.9835**	0.7671	**0.8716**	**0.9790**
	STARE	**0.9672**	**0.9841**	0.7953	**0.8516**	**0.9883**

As is observed on table 5.3, the result of this implementation on the DRIVE dataset outperforms the state of the art result when considering the accuracy and AUC. The characteristics of the network in conjugation with the use of pre-processed data yielded very positive results, as the majority of the smaller and terminal vessels were segmented, the larger vessels were delineated with detail and background noise was heavily reduced when compared with the previously presented results. In Figure 5.6a is presented an original image from the DRIVE dataset, its manual segmentation is presented in figure 5.6b and the result of this implementation is visible on 5.6c.

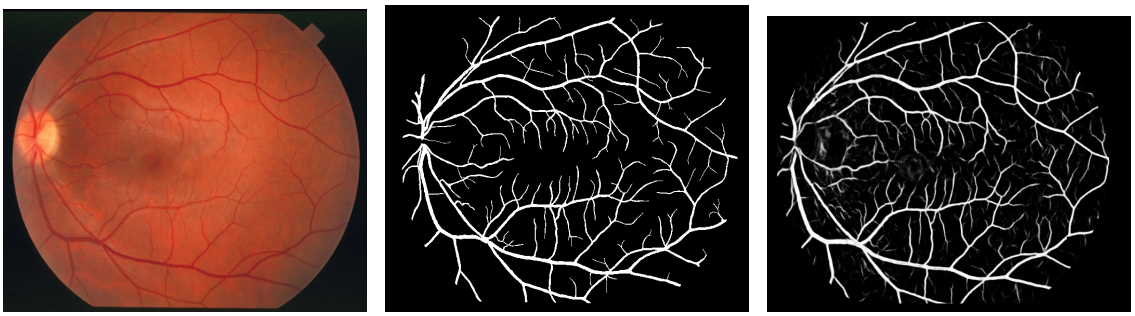
[1] Source: https://github.com/orobix/retina-unet



(a) (b) (c)

Figure 5.6: Original DRIVE image (a), its manual segmentation (b) and Best overall result of the U-Net based approach on DRIVE dataset. (c).

When considered the STARE dataset, the results are worse than those presented by Liskowski and Krawiec [51], but it is important to consider the division of the dataset mentioned on 5.1.1.1 and how the fewer images used to train and test the implementation can influence results. An original image from the STARE dataset, its manual segmentation and the final result are present in figures 5.7a- 5.7c.



(a) (b) (c)

Figure 5.7: Original STARE image (a), its manual segmentation (b) and Best overall result of the U-Net based approach on STARE dataset. (c).

5.2 Results of CNN for deep feature extraction

In this section are presented the results based on the methodology of work described on 4.3.

5.2.1 Random Forests

As described on previous chapter, to extract the features needed to train the classifiers was used a modified version of the LeNet architecture. The network was also was used to perform patch classification. The detailed results of the implementation of this network are presented on A.11. In this approach was only used the DRIVE dataset, since the limitations of the STARE dataset (described on 5.1.1.1) and the high number of features caused the implementation time to increase dramatically.

The training and feature extraction process took about 4 hours on a NVIDIA GeForce GT 650M graphics card. The training time when compared to the networks mentioned before is less due to the small size and complexity of this network.

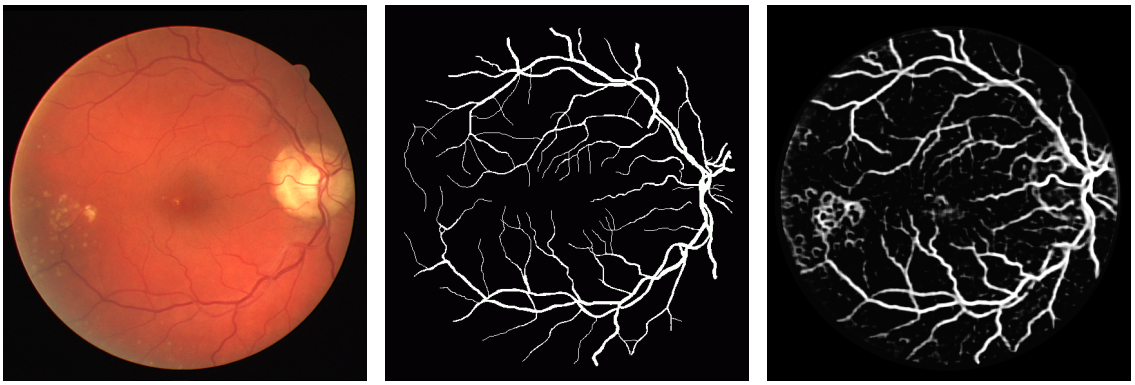
Regarding the training of the Random Forest classifier, it took considerably more time, about 4 days to complete. This was carried out using a NVIDIA GeForce GTX 1080ti graphics card. The results of each classifier and the different types of ensemble used are presented in detail on table A.12. The average results of each classifier and ensemble method used are presented on table 5.4.

Table 5.4: Average results of CNN, Random Forest Classifiers and Ensemble mechanism obtained on DRIVE dataset.

Algorithm	Dataset	Acc	SP	Sn	Pr	AUC
CNN	DRIVE	**0.9137**	0.9189	0.8816	0.6266	0.9578
RF01		0.9183	0.9162	0.8598	0.6225	0.9549
RF02		0.9093	0.9070	0.8450	0.6222	0.9473
RF03		0.8993	0.9052	0.8638	0.5871	0.9476
Max		0.8816	0.8800	**0.8988**	0.5443	0.8868
Min		**0.9249**	**0.9403**	0.8248	**0.6848**	0.9552
Mean		0.9101	0.9174	0.8660	0.6252	**0.9587**
Fuzzy-based Mecanism		0.9083	0.9162	0.8598	0.6225	0.8880

Considering the results presented, the best performances, considering only the classifiers and the CNN, are yielded from the CNN and the RF01 classifier trained with low dimensional features. Although the higher-order features learned from successive layers of a CNN are more abstract and therefore more suitable for object or image classification, in these particular implementation the best results are yielded from more elementary features.

In figure 5.8c is presented the best overall result, considering only the classifiers and the CNN. The original image and its manual segmentation are presented on figure 5.8a and 5.8b.

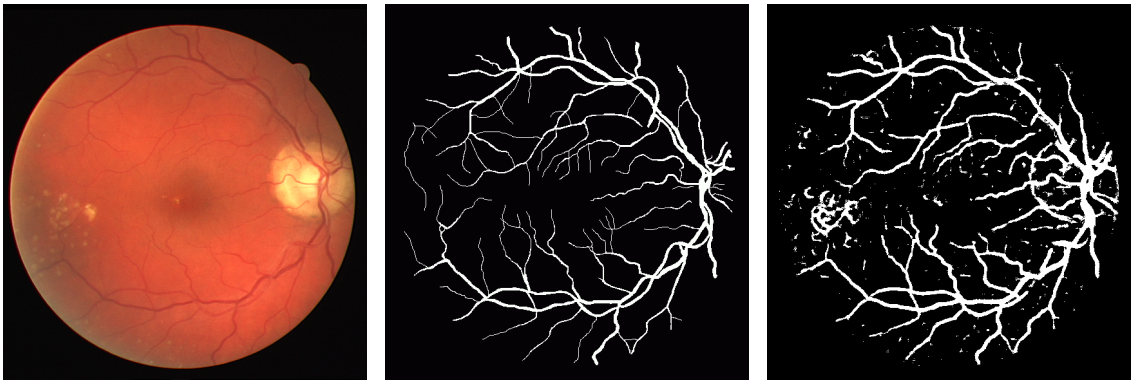


(a) (b) (c)

Figure 5.8: Original DRIVE image (a), its manual segmentation (b) and Best overall result of CNN and Random Forest classifiers of this approach on DRIVE dataset (c).

Considering now the ensemble mechanism, the best results are yielded when are used the minimum prediction of the three classifiers. Using this ensemble method, the number of false positive is reduced and with this some of the background noise are eliminated. The final binary segmentation is presented on figure 5.9c . The original image and it's manual segmentation are presented on figure 5.9a and 5.9b.

Comparing now the obtained results with the state of the art, more specifically, with the original work [44], the presented implementation under-performed across all the metrics in analysis. When compared with other deep learning approach, this implementation present results close to the ones present by Melinscak *et al.* [48], but not as good, considering the accuracy and AUC.



(a) (b) (c)

Figure 5.9: Original DRIVE image (a), its manual segmentation (b) and Best ensemble result of this approach on DRIVE dataset (c).

5.2.2 Support Vector Machines

Considering now the classification process using the SVM classifiers, the overall feature extraction process was the same as the one used for the RF classifier, with the addition of the feature selection mechanisms.

As described before, three different implementation were tested. In first one, all the features extracted from the CNN were used to train the SVM classifiers, and it took about 5 days to complete using the same graphics card. The approach using Univariate features took approximately the same number of days as the one mentioned before. The last implementation tested was the one using recursive feature selection with cross validation, which took at least 8 day to complete due to the size of the feature dataset used.

Table 5.5: Average results of CNN and the different SVM classifiers used obtained on DRIVE dataset.

Algorithm	Dataset	Acc	SP	Sn	Pr	AUC
CNN	DRIVE	**0.9138**	**0.9189**	**0.8816**	**0.6265**	**0.9578**
SVM		0.5631	0.5741	0.4900	0.1479	0.5321
SVM - Univariate Selection		0.7072	0.7405	0.4871	0.2207	0.6138
SVM- RFECV		0.7825	0.8574	0.2865	0.2326	0.5720

The results of these implementations are summarized on table 5.5, these are the best preforming ones considering the three classifiers and all the different ensemble mechanisms. Considering the presented results,it is clear that they are unacceptable when compared with all the implementations analyzed before and with those presented on table 3.6.

Since the results obtained were considerably low, different margin values as well as different hyperparametres were tested, but the results remained as unacceptable as the ones obtained in the first tests.

Chapter 6

Conclusions and Future work

This chapter is reserved for final conclusions of the work done and possibilities of future work in aspects of optimizations and additional functionalities.

6.1 Conclusions

The work of this thesis was focused on the study and interpretation of how deep learning can be used in the task of segmentation of retina blood vessels. To understand how this process works and which aspects can influence its performance and results, two main methodologies were implemented, tested and analyzed.

Considering the more traditional approach, where a CNN is used for patch classification and segmentation, different networks were implemented to assess the correlation between the complexity of the network, training process, and results. As expected, the results have showed that the use of a more complex network yielded the best set of results in patch classification, when compared to a simpler architecture such as LeNet. The training process is also influenced by the complexity of the network, as more complex networks take considerably more time. The training process can also be improved by using dropout between the fully connected layers of the network and by normalizing the network input data. The experiments have shown that these two mechanisms induced lower training times and larger accuracies on the test set.

Also regarding the classification task, its performance is considerably improved when the surrounding information of the central pixel is taken into consideration (structure prediction). The use of structure prediction, in this particular case increased the average accuracy from 0.9101 to 0.9434 on the same set of images.

The use of pre-processed images also improved the results, as a higher number of thinner vessels and other details of the vascular tree become more apparent after correcting some non-uniformities of the images and enhancing small structures. As some non-uniformity of the RGB image are eliminated during the process. When comparing the results of the same Liskowski based implementation trained with different data, the results showed an increase, even though very subtle

way, of the accuracy and AUC (when compared the results from the Liskowski SP with Liskwoski SP RGB).

From the analysis of the results of these more classic approaches it is possible to concluded that some details and parameters, such as the learning rate and the optimizer used, can influence the results. As some of these are not disclosed by the authors, some discrepancies between the obtained results and the ones presented on the state of the art exist.

Considering now the methodologies that use the CNN as a deep feature extractor: when comparing to a classic approach, the large number of samples used is a setback when these implementations are taken into consideration. The computation time and the necessity of a feature selection mechanism considering the high number of samples can affect the practicality of these solutions.

The best performance obtained in this thesis was based on the u-net approach, in this were used pre-processed images and a more complex and robust architecture was considered in the training process. From these it is also possible to assess how important these aspects can for the improvement of metrics such as accuracy and AUC, when compared with simpler networks and the type of data used. Comparing the obtained results to those presented on the state of the art, this implementation yielded comparable results to the best one presented regarding accuracy and AUC on the DRIVE dataset.

From the results of the Random forest implementations, the best performances, are yielded from the classifier trained with lower dimensional features. Although the higher-order features learned from successive layers of a CNN are more abstract and therefore more suitable for object or image classification, in this particular implementation the best results are yielded from more elementary features.

Considering the weak set of results obtained in the different implementations of the SVM based approach, it is not possible to conclude anything about their performance, and the importance of feature selection mechanisms in this classifier. Also, is important to mention that the performance of these implementations were influenced by the number of samples used, as the number of folds in the recursive feature selection should be at least 3 to 5, but due to number of samples used, the computational effort required is extremely high.

6.2 Future Work

Regarding all the different approaches and the set of results obtained, there are some aspects that can be improved.

Regarding the patch classification based approaches, these can be improved with the additional study of tuning techniques and parameters such as the optimizers used and weight initialization. In addition to this, the use of augmented data to train these implementations can also be applied to improve their final results.

Since the u-net based approach is the one that yielded the best results, in the future it would be interesting to use this on different databases such as CHASE and ARIA, in order to assess the performance of the solution when different data are used.

Considering now the work done regarding the use of CNN for deep features extraction, the process of feature extraction can be improved using a more complex and robust network such as VGG16. Also the use of one of the available pre-trained networks for this task can save time and improve the performance.

As mentioned before, the classification process required a considerable amount of time, one key aspect to improve this is the use of feature selection mechanism applied to all the implementations. Regarding the SVM based implementations, the results obtained were not the expected, as future work, repeating these implementations and applying fine tuning techniques should be considered. The feature selection mechanism can also be enhanced by increasing the number of folds used for cross validation (to 5 or at least 3), in the case of the recursive feature selection with cross-validation.

Appendix A

Tables of results

A.1 Results of CNN Based approaches for retinal vessel segmentation

A.1.1 Results of the Melinscak based approach

Table A.1: Results of the Melinscak based approach tested and trained on DRIVE dataset

Algorithm	Dataset	Img	Acc	SP	Sn	Pr	AUC
Melinscak	DRIVE	1	0.9180	0.9186	0.9144	0.6288	0.971
		2	0.9267	0.9276	0.9215	0.6918	0.9735
		3	0.9175	0.9327	0.8281	0.6773	0.9517
		4	0.9315	0.9398	0.8779	0.6915	0.9604
		5	0.9344	0.9487	0.8439	0.7205	0.9507
		6	0.9295	0.9521	0.7919	0.7307	0.945
		7	0.9269	0.9380	0.8541	0.6776	0.9475
		8	0.9312	0.9483	0.8121	0.6932	0.9546
		9	0.9274	0.9360	0.8628	0.6421	0.9588
		10	0.9085	0.9090	0.9049	0.5741	0.962
		11	0.8809	0.8772	0.9058	0.5234	0.9545
		12	0.9076	0.9067	0.9141	0.5836	0.9646
		13	0.9142	0.9185	0.8883	0.6429	0.9602
		14	0.8927	0.8862	0.9418	0.5249	0.9698
		15	0.8454	0.8216	0.9541	0.3826	0.9626
		16	0.9114	0.9106	0.9174	0.6069	0.9686
		17	0.9180	0.9245	0.8714	0.6189	0.9605
		18	0.9066	0.9036	0.9301	0.5559	0.9648
		19	0.8903	0.8811	0.9575	0.5242	0.9745
		20	0.9078	0.9030	0.9477	0.5384	0.9546
		Mean	0.9101	0.9142	0.8920	0.6115	0.9604

Table A.2: Results of the Melinscak based approach tested and trained on STARE dataset

Algorithm	Dataset	Img	Acc	SP	Sn	Pr	AUC
Melinscak	STARE	1	0.9192	0.9884	0.5145	0.8839	0.9259
		2	0.9315	0.9521	0.7889	0.7051	0.9468
		3	0.9600	0.9957	0.5135	0.9052	0.952
		4	0.9671	0.9940	0.4944	0.8242	0.9479
		5	0.9343	0.9930	0.3839	0.8540	0.9150
		Mean	0.9424	0.9847	0.5390	0.8345	0.93752

A.1.2 Results of Liskowski based approachs

Table A.3: Results of the Liskowski based approach Trained on DRIVE dataset

Algorithm	Dataset	Img	Acc	SP	Sn	Pr	AUC
Liskowski SP	DRIVE	1	0.9476	0.9500	0.8560	0.7207	0.9747
		2	0.9481	0.9594	0.8840	0.7933	0.9812
		3	0.9455	0.9688	0.7750	0.8089	0.9632
		4	0.9519	0.9733	0.8124	0.8242	0.9654
		5	0.9494	0.9767	0.7759	0.8392	0.9624
		6	0.9415	0.9791	0.7124	0.8487	0.9552
		7	0.9470	0.9722	0.7815	0.8111	0.9638
		8	0.9437	0.9758	0.7200	0.8107	0.9607
		9	0.9500	0.9688	0.7997	0.7736	0.9652
		10	0.9382	0.9483	0.8640	0.6938	0.9684
		11	0.9212	0.9259	0.8668	0.6354	0.9538
		12	0.9370	0.9439	0.8891	0.6939	0.9773
		13	0.9481	0.9538	0.8429	0.7510	0.9688
		14	0.9293	0.9306	0.9196	0.6390	0.9760
		15	0.9099	0.8975	0.8895	0.4569	0.9437
		16	0.9386	0.9475	0.8792	0.7159	0.9753
		17	0.9408	0.9586	0.8146	0.7343	0.9667
		18	0.9455	0.9365	0.8841	0.6438	0.9723
		19	0.9296	0.9250	0.9250	0.6312	0.9815
		20	0.9460	0.9392	0.9092	0.6411	0.9785
		Mean	0.9434	0.9515	0.8400	0.7233	0.9677

Table A.4: Results of the Liskowski based approach trained using balanced data on DRIVE dataset

Algorithm	**Dataset**	**Img**	**Acc**	**SP**	**Sn**	**Pr**	**AUC**
Liskowski SP Balanced	DRIVE	1	0.9209	0.9284	0.8718	0.6473	0.9667
		2	0.9379	0.9486	0.8768	0.7505	0.9726
		3	0.9195	0.9874	0.5215	0.8758	0.9277
		4	0.9310	0.9417	0.8613	0.6945	0.9583
		5	0.9420	0.9687	0.7718	0.7948	0.9505
		6	0.9345	0.9635	0.7574	0.7734	0.9501
		7	0.9446	0.9476	0.7863	0.6959	0.9430
		8	0.9374	0.9411	0.7525	0.6475	0.9364
		9	0.9439	0.9774	0.6920	0.8028	0.9488
		10	0.9235	0.9333	0.8507	0.6338	0.9548
		11	0.9189	0.9342	0.8167	0.6488	0.9463
		12	0.9302	0.9424	0.8455	0.6773	0.9604
		13	0.9286	0.9503	0.7975	0.7259	0.9521
		14	0.9250	0.9308	0.8822	0.6301	0.9668
		15	0.9322	0.9310	0.8124	0.5165	0.9629
		16	0.9314	0.9537	0.7830	0.7180	0.9603
		17	0.9450	0.9699	0.6870	0.7625	0.9476
		18	0.9396	0.9421	0.8336	0.6512	0.9642
		19	0.8971	0.8939	0.9205	0.5427	0.9012
		20	0.9372	0.9452	0.8603	0.6132	0.9688
		Mean	0.9310	0.9466	0.7990	0.6901	0.9520

Table A.5: Results of the Liskowski based approach trained using random data on DRIVE dataset

Algorithm	**Dataset**	**Img**	**Acc**	**SP**	**Sn**	**Pr**	**AUC**
Liskowski SP Random	DRIVE	1	0.9376	0.9586	0.7978	0.7442	0.9613
		2	0.9471	0.9829	0.7436	0.8848	0.9633
		3	0.9281	0.9874	0.5805	0.8868	0.9416
		4	0.9454	0.9770	0.7397	0.8319	0.9507
		5	0.9389	0.9898	0.6142	0.9047	0.9399
		6	0.9381	0.9836	0.6611	0.8689	0.9417
		7	0.9386	0.9759	0.6942	0.8147	0.9347
		8	0.9425	0.9772	0.6217	0.7968	0.9337
		9	0.9410	0.9826	0.5534	0.9090	0.9411
		10	0.9430	0.9730	0.7215	0.7838	0.9464
		11	0.9378	0.9696	0.7245	0.7802	0.9367
		12	0.9447	0.9766	0.7212	0.8154	0.9534
		13	0.9481	0.9806	0.6803	0.8530	0.9445
		14	0.9455	0.9705	0.7584	0.7748	0.9564
		15	0.9441	0.9646	0.7670	0.7154	0.9523
		16	0.9398	0.9791	0.6784	0.8304	0.9518
		17	0.9378	0.9761	0.5938	0.8576	0.9406
		18	0.9431	0.9725	0.7165	0.7720	0.9548
		19	0.9431	0.9567	0.8438	0.7272	0.9644
		20	0.9461	0.9665	0.7340	0.7340	0.9677
		Mean	0.9415	0.9751	0.6973	0.8143	0.9489

Table A.6: Results of the Liskowski based approach trained on RGB data From DRIVE dataset

Algorithm	Dataset	Img	Acc	SP	Sn	Pr	AUC
Liskowski SP RGB	DRIVE	1	0.9376	0.9586	0.7978	0.7442	0.9548
		2	0.9471	0.9829	0.7436	0.8848	0.9463
		3	0.9281	0.9874	0.5805	0.8868	0.9604
		4	0.9454	0.9770	0.7397	0.8319	0.9259
		5	0.9389	0.9898	0.6142	0.9047	0.9468
		6	0.9381	0.9836	0.6611	0.8689	0.961
		7	0.9386	0.9759	0.6942	0.8147	0.9304
		8	0.9325	0.9772	0.6217	0.7968	0.9409
		9	0.9410	0.9726	0.5534	0.9090	0.9474
		10	0.9430	0.9730	0.7215	0.7838	0.9506
		11	0.9378	0.9696	0.7245	0.7802	0.9416
		12	0.9447	0.9766	0.7212	0.8154	0.9538
		13	0.9381	0.9806	0.6803	0.8530	0.9773
		14	0.9455	0.9705	0.7584	0.7748	0.9716
		15	0.9441	0.9646	0.7670	0.7154	0.974
		16	0.9398	0.9791	0.6784	0.8304	0.9512
		17	0.9378	0.9861	0.5938	0.8576	0.9653
		18	0.9431	0.9725	0.7165	0.7165	0.9661
		19	0.9431	0.9567	0.8438	0.7272	0.9352
		20	0.9461	0.9665	0.7756	0.7340	0.9425
		Mean	0.9405	0.9751	0.6994	0.8115	0.9522

Table A.7: Results of the Liskowski based approach trained on STARE dataset

Algorithm	Dataset	Img	Acc	SP	Sn	Pr	AUC
Lisk sp	STARE	1	0.9476	0.9500	0.8560	0.7207	0.9630
		2	0.9481	0.9594	0.8840	0.7933	0.9536
		3	0.9455	0.9688	0.7750	0.8089	0.9458
		4	0.9519	0.9733	0.8124	0.8242	0.9674
		5	0.9494	0.9767	0.7759	0.8392	0.9528
		Mean	0.9485	0.9656	0.8207	0.7973	0.9565

Table A.8: Results of the Liskowski based approach trained using balanced data on STARE dataset

Algorithm	Dataset	Img	Acc	SP	Sn	Pr	AUC
Lisk sp Balanced	STARE	1	0.9366	0.9456	0.8413	0.7303	0.9610
		2	0.9368	0.9596	0.8795	0.7746	0.9436
		3	0.9351	0.9579	0.7622	0.8195	0.9517
		4	0.9486	0.9748	0.8053	0.8368	0.9705
		5	0.9384	0.9726	0.7633	0.8457	0.9479
		Mean	0.9391	0.9621	0.8103	0.8014	0.9549

Table A.9: Results of the Liskowski based approach trained using random data on STARE dataset

Algorithm	**Dataset**	**Img**	**Acc**	**SP**	**Sn**	**Pr**	**AUC**
Lisk sp Random	STARE	1	0.9421	0.9486	0.8360	0.7152	0.9563
		2	0.9502	0.9578	0.8523	0.8127	0.9624
		3	0.9403	0.9643	0.7524	0.7951	0.9484
		4	0.9489	0.9823	0.7856	0.8185	0.9529
		5	0.9437	0.9689	0.7902	0.8214	0.9587
		Mean	0.9450	0.9644	0.8033	0.7926	0.9557

Table A.10: Results of the Liskowski based approach trained using RGB Data from STARE dataset

Algorithm	**Dataset**	**Img**	**Acc**	**SP**	**Sn**	**Pr**	**AUC**
Lisk sp RGB	STARE	1	0.9436	0.9321	0.8420	0.7692	0.9612
		2	0.9301	0.9458	0.8821	0.7847	0.9425
		3	0.9349	0.9732	0.7681	0.7952	0.9320
		4	0.9545	0.9701	0.8374	0.8127	0.9657
		5	0.9503	0.9740	0.7985	0.8294	0.9762
		Mean	0.9427	0.9590	0.8256	0.7982	0.9555

A.2 Results of CNN for deep feature extraction approaches

Table A.11: Results of CNN used for feature extraction trained on DRIVE dataset

Algorithm	Dataset	Img	Acc	SP	Sn	Pr	AUC
Wang	DRIVE	1	0.9235	0.9235	0.9060	0.6493	0.9726
		2	0.9301	0.9332	0.9129	0.7065	0.9756
		3	0.9273	0.9435	0.8324	0.7153	0.9592
		4	0.9365	0.9480	0.8617	0.7181	0.9618
		5	0.9371	0.9535	0.8327	0.7378	0.9574
		6	0.9342	0.9613	0.7692	0.7656	0.9524
		7	0.9332	0.9483	0.8349	0.7112	0.9577
		8	0.9333	0.9538	0.7907	0.7112	0.9556
		9	0.9317	0.9430	0.8474	0.6642	0.9605
		10	0.9121	0.9142	0.8971	0.5864	0.9619
		11	0.8828	0.8809	0.8953	0.5283	0.953
		12	0.9080	0.9071	0.9139	0.5847	0.9639
		13	0.9170	0.9241	0.8744	0.6555	0.961
		14	0.8884	0.8807	0.9462	0.5146	0.9304
		15	0.8251	0.8114	0.9428	0.3668	0.9409
		16	0.9162	0.9180	0.9045	0.6240	0.9386
		17	0.9247	0.9338	0.8607	0.6463	0.9389
		18	0.9132	0.9124	0.9198	0.5767	0.9682
		19	0.8892	0.8813	0.9476	0.5220	0.972
		20	0.9106	0.9068	0.9421	0.5469	0.9748
		Mean	0.9137	0.9189	0.8816	0.6266	0.95782

Table A.12: Complete results from the CNN. Random Forest classifiers and ensemble mechanism on DRIVE dataset.

Algorithm	Dataset	Img	Ensemble Method	Acc	SP	Sn	Pr	AUC
		1	RF01	0.9310	0.9406	0.8680	0.6878	0.9673
			RF02	0.9290	0.9360	0.8827	0.6753	0.97
			RF03	0.9197	0.9263	0.8758	0.6420	0.9635
			Max	0.9133	0.9134	0.9125	0.6138	0.9129
			Min	0.9374	0.9532	0.8326	0.7286	0.9686
			Mean	0.9295	0.9369	0.8804	0.6778	0.9702
			Emsemble Mecanism	0.9310	0.9406	0.8680	0.6878	0.9043
		2	RF01	0.9372	0.9463	0.8855	0.7440	0.9696
			RF02	0.9361	0.9433	0.8952	0.7357	0.9726
			RF03	0.9253	0.9303	0.8973	0.6940	0.9681
			Max	0.9184	0.9176	0.9228	0.6638	0.9202
			Min	0.9444	0.9595	0.8587	0.7889	0.9715
			Mean	0.9353	0.9422	0.8958	0.7321	0.9732
			Emsemble Mecanism	0.9372	0.9463	0.8855	0.7440	0.9159
		3	RF01	0.9229	0.9412	0.8155	0.7030	0.9513
			RF02	0.9239	0.9432	0.8104	0.7088	0.9509
			RF03	0.9135	0.9325	0.8023	0.6694	0.9428
			Max	0.9034	0.9104	0.8620	0.6213	0.8862
			Min	0.9324	0.9628	0.7538	0.7755	0.9496
			Mean	0.9250	0.9445	0.8104	0.7135	0.9536
			Emsemble Mecanism	0.9229	0.9412	0.8155	0.7030	0.8784
		4	RF01	0.9401	0.9580	0.8239	0.7510	0.951
			RF02	0.9394	0.9546	0.8407	0.7403	0.956
			RF03	0.9308	0.9443	0.8433	0.6996	0.9516
			Max	0.9257	0.9336	0.8739	0.6695	0.9037
			Min	0.9453	0.9685	0.7942	0.7952	0.9549
			Mean	0.9393	0.9546	0.8401	0.7400	0.9564
			Emsemble Mecanism	0.9401	0.9580	0.8239	0.7510	0.891
		5	RF01	0.9412	0.9640	0.7957	0.7763	0.946
			RF02	0.9398	0.9597	0.8128	0.7600	0.9501
			RF03	0.9298	0.9475	0.8173	0.7097	0.9447
			Max	0.9266	0.9386	0.8501	0.6848	0.8943

			Min	0.9448	0.9731	0.7645	0.8172	0.9493
			Mean	0.9394	0.9596	0.8103	0.7591	0.9506
			Emsemble Mecanism	0.9412	0.9640	0.7957	0.7763	0.8798
		6	RF01	0.9372	0.9714	0.7287	0.8074	0.9429
			RF02	0.9358	0.9685	0.7368	0.7936	0.9438
			RF03	0.9289	0.9595	0.7430	0.7508	0.9334
			Max	0.9282	0.9513	0.7877	0.7265	0.8695
			Min	0.9375	0.9793	0.6830	0.8443	0.9421
			Mean	0.9361	0.9690	0.7361	0.7959	0.9464
			Emsemble Mecanism	0.9372	0.9714	0.7287	0.8074	0.8501
		7	RF01	0.9301	0.9490	0.8067	0.7070	0.9453
			RF02	0.9324	0.9498	0.8181	0.7132	0.9488
			RF03	0.9204	0.9355	0.8215	0.6601	0.9434
			Max	0.9106	0.9191	0.8548	0.6173	0.887
			Min	0.9407	0.9663	0.7732	0.7779	0.9479
			Mean	0.9311	0.9484	0.8181	0.7075	0.9502
			Emsemble Mecanism	0.9301	0.9490	0.8067	0.7070	0.8779
		8	RF01	0.9291	0.9593	0.7188	0.7175	0.9384
			RF02	0.9311	0.9575	0.7473	0.7168	0.9434
			RF03	0.9228	0.9479	0.7484	0.6737	0.937
			Max	0.9160	0.9328	0.7994	0.6308	0.8661
			Min	0.9357	0.9737	0.6710	0.7860	0.9432
			Mean	0.9313	0.9586	0.7415	0.7202	0.9455
			Emsemble Mecanism	0.9291	0.9593	0.7188	0.7175	0.839
		9	RF01	0.9345	0.9486	0.8286	0.6821	0.9556
			RF02	0.9327	0.9467	0.8274	0.6739	0.9543
			RF03	0.9184	0.9308	0.8246	0.6134	0.9462
			Max	0.9112	0.9170	0.8678	0.5818	0.8924
			Min	0.9420	0.9629	0.7849	0.7380	0.9524
			Mean	0.9318	0.9456	0.8279	0.6694	0.9567
			Emsemble Mecanism	0.9345	0.9486	0.8286	0.6821	0.8886
		10	RF01	0.9004	0.9020	0.8884	0.5515	0.955
			RF02	0.9024	0.9046	0.8866	0.5575	0.9542
			RF03	0.8939	0.8959	0.8792	0.5338	0.9488
			Max	0.8684	0.8619	0.9167	0.4738	0.8893
			Min	0.9246	0.9346	0.8507	0.6383	0.953

	Mean	0.9045	0.9071	0.8852	0.5638	0.9567
	Emsemble Mecanism	0.9004	0.9020	0.8884	0.5515	0.8952
11	RF01	0.8627	0.8583	0.8920	0.4839	0.9425
	RF02	0.8734	0.8707	0.8916	0.5066	0.9437
	RF03	0.8606	0.8566	0.8872	0.4797	0.9384
	Max	0.8228	0.8081	0.9214	0.4170	0.8648
	Min	0.9016	0.9083	0.8569	0.5818	0.9424
	Mean	0.8734	0.8705	0.8931	0.5067	0.9464
	Emsemble Mecanism	0.8627	0.8583	0.8920	0.4839	0.8752
12	RF01	0.8948	0.8925	0.9107	0.5480	0.9607
	RF02	0.8992	0.8978	0.9092	0.5600	0.961
	RF03	0.8892	0.8875	0.9010	0.5340	0.9531
	Max	0.8617	0.8508	0.9380	0.4736	0.8943
	Min	0.9213	0.9282	0.8730	0.6350	0.959
	Mean	0.9006	0.8995	0.9082	0.5638	0.9635
	Emsemble Mecanism	0.8948	0.8925	0.9107	0.5480	0.9016
13	RF01	0.9159	0.9256	0.8572	0.6555	0.9545
	RF02	0.9162	0.9253	0.8608	0.6557	0.9552
	RF03	0.9025	0.9096	0.8595	0.6109	0.9492
	Max	0.8884	0.8865	0.8998	0.5670	0.8931
	Min	0.9296	0.9482	0.8167	0.7227	0.9543
	Mean	0.9167	0.9258	0.8619	0.6573	0.957
	Emsemble Mecanism	0.9159	0.9256	0.8572	0.6555	0.8914
14	RF01	0.8723	0.8631	0.9404	0.4788	0.9643
	RF02	0.8774	0.8694	0.9370	0.4896	0.963
	RF03	0.8682	0.8598	0.9311	0.4702	0.9583
	Max	0.8337	0.8169	0.9596	0.4120	0.8353
	Min	0.9056	0.9048	0.9110	0.5613	0.9653
	Mean	0.8789	0.8710	0.9377	0.4929	0.9661
	Emsemble Mecanism	0.8723	0.8631	0.9404	0.4788	0.9018
15	RF01	0.7953	0.7797	0.9297	0.3284	0.9393
	RF02	0.7862	0.7688	0.9361	0.3193	0.9426
	RF03	0.7920	0.7767	0.9246	0.3242	0.9362
	Max	0.7237	0.6966	0.9574	0.2677	0.827
	Min	0.8531	0.8477	0.8999	0.4065	0.9431

			Mean	0.7995	0.7840	0.9334	0.3336	0.9456
			Emsemble Mecanism	0.7953	0.7797	0.9297	0.3284	0.8547
		16	RF01	0.9126	0.9160	0.8906	0.6147	0.9622
			RF02	0.9154	0.9200	0.8848	0.6247	0.9621
			RF03	0.9031	0.9068	0.8786	0.5866	0.9551
			Max	0.8869	0.8820	0.9194	0.5399	0.9007
			Min	0.9293	0.9414	0.8490	0.6857	0.9613
			Mean	0.9147	0.9193	0.8841	0.6226	0.9636
			Emsemble Mecanism	0.9126	0.9160	0.8906	0.6147	0.9033
		17	RF01	0.9168	0.9289	0.8308	0.6217	0.9474
			RF02	0.9208	0.9334	0.8310	0.6372	0.9506
			RF03	0.9104	0.9226	0.8242	0.5995	0.9416
			Max	0.8944	0.8971	0.8748	0.5446	0.8865
			Min	0.9328	0.9542	0.7806	0.7057	0.9481
			Mean	0.9215	0.9346	0.8283	0.6403	0.9521
			Emsemble Mecanism	0.9168	0.9289	0.8308	0.6217	0.8798
		18	RF01	0.9046	0.9038	0.9103	0.5512	0.9632
			RF02	0.9114	0.9118	0.9079	0.5720	0.966
			RF03	0.9016	0.9015	0.9022	0.5432	0.9601
			Max	0.8794	0.8720	0.9365	0.4870	0.9041
			Min	0.9272	0.9341	0.8744	0.6326	0.9641
			Mean	0.9110	0.9113	0.9083	0.5706	0.9668
			Emsemble Mecanism	0.9046	0.9038	0.9103	0.5512	0.907
		19	RF01	0.8806	0.8718	0.9448	0.5021	0.9705
			RF02	0.8846	0.8846	0.9423	0.5111	0.9699
			RF03	0.8760	0.8679	0.9356	0.4921	0.9655
			Max	0.8429	0.8266	0.9614	0.4314	0.894
			Min	0.9121	0.9111	0.9191	0.5860	0.9697
			Mean	0.8869	0.8794	0.9420	0.5166	0.9724
			Emsemble Mecanism	0.8806	0.8718	0.9448	0.5021	0.9083
		20	RF01	0.9072	0.9044	0.9304	0.5374	0.9716
			RF02	0.9070	0.9029	0.9408	0.5364	0.974
			RF03	0.8945	0.8894	0.9373	0.5029	0.9688
			Max	0.8770	0.8671	0.9600	0.4631	0.9143
			Min	0.9250	0.9269	0.9091	0.5975	0.9733
			Mean	0.9067	0.9027	0.9397	0.5355	0.9747

			Emsemble Mecanism	0.9072	0.9044	0.9304	0.5374	0.9174

References

[1] Michael D. Abramoff, Mona K. Garvin, and Milan Sonka. Retinal imaging and image analysis. *IEEE Reviews in Biomedical Engineering*, 3:169–208, 2010. doi:10.1109/rbme.2010.2084567.

[2] Nicolas M. Ducrey. Monochromatic ophthalmoscopy and fundus photography. *Archives of Ophthalmology*, 97(2):288, 1979. doi:10.1001/archopht.1979.01020010140009.

[3] M.M. Fraz, P. Remagnino, A. Hoppe, B. Uyyanonvara, A.R. Rudnicka, C.G. Owen, and S.A. Barman. Blood vessel segmentation methodologies in retinal images – a survey. *Computer Methods and Programs in Biomedicine*, 108(1):407–433, 2012. doi:10.1016/j.cmpb.2012.03.009.

[4] June-Goo Lee, Sanghoon Jun, Young-Won Cho, Hyunna Lee, Guk Bae Kim, Joon Beom Seo, and Namkug Kim. Deep learning in medical imaging: General overview. *Korean Journal of Radiology*, 18(4):570, 2017. doi:10.3348/kjr.2017.18.4.570.

[5] Göran Darius Hildebrand and Alistair R. Fielder. Anatomy and physiology of the retina. *Pediatric Retina*, pages 39–65, 2010. doi:10.1007/978-3-642-12041-1_2.

[6] Barbara Blodi. Macular and retinal diseases: Recent advances in diagnosis and therapy: Developments in ophthalmology. *Archives of Ophthalmology*, 117(2):289, 1999. doi:10.1001/archopht.117.2.289.

[7] H. R Taylor. World blindness: a 21st century perspective. *British Journal of Ophthalmology*, 85(3):261–266, 2001. doi:10.1136/bjo.85.3.261.

[8] Barry R. Masters. Kanski's clinical ophthalmology, a systematic approach. eighth edition brad bowling (2016) 917pp., 2,600 illustrations isbn: 9780702055720 elsevier. *Graefe's Archive for Clinical and Experimental Ophthalmology*, 255(9):1867–1868, 2016. doi:10.1007/s00417-016-3549-x.

[9] H. Wassle and B. B. Boycott. Functional architecture of the mammalian retina. *Physiological Reviews*, 71(2):447–480, 1991. doi:10.1152/physrev.1991.71.2.447.

[10] Marena Patronas, Arnold J. Kroll, Peter L. Lou, and Edward A. Ryan. A review of vitreoretinal interface pathology. *International Ophthalmology Clinics*, 49(1):133–143, 2009. doi:10.1097/iio.0b013e3181924b3e.

[11] José G. Cunha-Vaz. The blood–retinal barriers system. basic concepts and clinical evaluation. *Experimental Eye Research*, 78(3):715–721, 2004. doi:10.1016/s0014-4835(03)00213-6.

[12] Howard C. Howland. The human eye: Structure and function. clyde w. oyster. *The Quarterly Review of Biology*, 75(1):90–90, 2000. doi:10.1086/393359.

[13] Lawrence A. Yannuzzi, Michael D. Ober, Jason S. Slakter, Richard F. Spaide, Yale L. Fisher, Robert W. Flower, and Richard Rosen. Ophthalmic fundus imaging: today and beyond. *American Journal of Ophthalmology*, 137(3):511–524, 2004. doi:10.1016/j.ajo.2003.12.035.

[14] Luca Giancardo. Automated fundus images analysis techniques to screen retinal diseases in diabetic patients, 2018. URL: https://tel.archives-ouvertes.fr/tel-00692354.

[15] M. M. Fraz, P. Remagnino, and A. Hoppe. A supervised method for retinal blood vessel segmentation using line strength, multiscale gabor and morphological features. *2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pages 410–5, 2011.

[16] Oliver Faust, Rajendra Acharya U., E. Y. Ng, Kwan-Hoong Ng, and Jasjit S. Suri. Algorithms for the automated detection of diabetic retinopathy using digital fundus images: A review. *J. Med. Syst.*, 36(1):145–157, February 2012. URL: http://dx.doi.org/10.1007/s10916-010-9454-7, doi:10.1007/s10916-010-9454-7.

[17] R.J. Winder, P.J. Morrow, I.N. McRitchie, J.R. Bailie, and P.M. Hart. *Computerized Medical Imaging and Graphics*, 33(8):608–622, 2009. doi:10.1016/j.compmedimag.2009.06.003.

[18] Sameh A. Salem, Nancy M. Salem, and Asoke K. Nandi. Segmentation of retinal blood vessels using a novel clustering algorithm with a partial supervision strategy. *Medical & Biological Engineering & Computing*, 45(3):261–273, 2007. doi:10.1007/s11517-006-0141-2.

[19] Giri Babu Kande, P. Venkata Subbaiah, and T. Satya Savithri. Unsupervised fuzzy based vessel segmentation in pathological digital fundus images. *Journal of Medical Systems*, 34(5):849–858, 2009. doi:10.1007/s10916-009-9299-0.

[20] Fabiola M. Villalobos-Castaldi, Edgardo M. Felipe-Riverón, and Luis P. Sánchez-Fernández. A fast, efficient and automated method to extract vessels from fundus images. *Journal of Visualization*, 13(3):263–270, 2010. doi:10.1007/s12650-010-0037-y.

[21] S. Chaudhuri, S. Chatterjee, N. Katz, M. Nelson, and M. Goldbaum. Detection of blood vessels in retinal images using two-dimensional matched filters. *IEEE Transactions on Medical Imaging*, 8(3):263–269, 1989. doi:10.1109/42.34715.

[22] A.D. Hoover, V. Kouznetsova, and M. Goldbaum. Locating blood vessels in retinal images by piecewise threshold probing of a matched filter response. *IEEE Transactions on Medical Imaging*, 19(3):203–210, 2000. doi:10.1109/42.845178.

[23] Bob Zhang, Lin Zhang, Lei Zhang, and Fakhri Karray. Retinal vessel extraction by matched filter with first-order derivative of gaussian. *Computers in Biology and Medicine*, 40(4):438–445, 2010. doi:10.1016/j.compbiomed.2010.02.008.

[24] Qin Li, Jane You, and David Zhang. Vessel segmentation and width estimation in retinal images using multiscale production of matched filter responses. *Expert Systems with Applications*, 39(9):7600–7610, 2012. doi:10.1016/j.eswa.2011.12.046.

[25] F. Zana and J.-C. Klein. Segmentation of vessel-like patterns using mathematical morphology and curvature evaluation. *IEEE Transactions on Image Processing*, 10(7):1010–1019, 2001. doi:10.1109/83.931095.

[26] G. Ayala, T. Leon, and V. Zapater. Different averages of a fuzzy set with an application to vessel segmentation. *IEEE Transactions on Fuzzy Systems*, 13(3):384–393, 2005. doi:10.1109/tfuzz.2004.839667.

[27] A.M. Mendonca and A. Campilho. Segmentation of retinal blood vessels by combining the detection of centerlines and morphological reconstruction. *IEEE Transactions on Medical Imaging*, 25(9):1200–1213, 2006. doi:10.1109/tmi.2006.879955.

[28] M S Miri and A Mahloojifar. Retinal image analysis using curvelet transform and multistructure elements morphology by reconstruction. *IEEE Transactions on Biomedical Engineering*, 58(5):1183–1192, 2011. doi:10.1109/tbme.2010.2097599.

[29] J.V.B. Soares, J.J.G. Leandro, R.M. Cesar, H.F. Jelinek, and M.J. Cree. Retinal vessel segmentation using the 2-d gabor wavelet and supervised classification. *IEEE Transactions on Medical Imaging*, 25(9):1214–1222, 2006. doi:10.1109/tmi.2006.879967.

[30] Ali Can, Hong Shen, J. N. Turner, H. L. Tanenbaum, and B. Roysam. Rapid automated tracing and feature extraction from retinal fundus images using direct exploratory algorithms. *Trans. Info. Tech. Biomed.*, 3(2):125–138, June 1999. URL: https://doi.org/10.1109/4233.767088, doi:10.1109/4233.767088.

[31] Marios Vlachos and Evangelos Dermatas. Multi-scale retinal vessel segmentation using line tracking. *Computerized Medical Imaging and Graphics*, 34(3):213–227, 2010. doi:10.1016/j.compmedimag.2009.09.006.

[32] Mouloud Adel, Aicha Moussaoui, Monique Rasigni, Salah Bourennane, and Latifa Hamami. Statistical-based tracking technique for linear structures detection: Application to vessel segmentation in medical images. *IEEE Signal Processing Letters*, 17(6):555–558, 2010. doi:10.1109/lsp.2010.2046697.

[33] Yi Yin, Mouloud Adel, and Salah Bourennane. Retinal vessel segmentation using a probabilistic tracking method. *Pattern Recognition*, 45(4):1235–1244, 2012. doi:10.1016/j.patcog.2011.09.019.

[34] Konstantinos K. Delibasis, Aristides I. Kechriniotis, C. Tsonos, and Nicholas Assimakis. Automatic model-based tracing algorithm for vessel segmentation and diameter estimation. *Computer Methods and Programs in Biomedicine*, 100(2):108–122, 2010. doi:10.1016/j.cmpb.2010.03.004.

[35] Ro F. Frangi, Koen L Vincken, W.J. Niessen, and Max A. Viergever. Multiscale vessel enhancement filtering,. *Lecture Notes in Computer Science*, page p. 130, 2000.

[36] M. Elena Martinez-Perez, Alun D. Hughes, Simon A. Thom, Anil A. Bharath, and Kim H. Parker. Segmentation of blood vessels from red-free and fluorescein retinal images. *Medical Image Analysis*, 11(1):47–61, 2007. doi:10.1016/j.media.2006.11.004.

[37] David E. Goldberg and John H. Holland. Genetic algorithms and machine learning. *Machine Learning*, 3(2):95–99, Oct 1988. URL: https://doi.org/10.1023/A:1022602019183, doi:10.1023/A:1022602019183.

[38] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117, 2015. URL: http://www.sciencedirect.com/science/article/pii/S0893608014002135, doi:https://doi.org/10.1016/j.neunet.2014.09.003.

[39] Guoqiang Zhong, Li-Na Wang, Xiao Ling, and Junyu Dong. An overview on data representation learning: From traditional feature learning to recent deep learning. *The Journal of Finance and Data Science*, 2(4):265–278, 2016. doi:10.1016/j.jfds.2017.05.001.

[40] Joes Staal, Michael D. Abràmoff, Meindert Niemeijer, Max A. Viergever, and Bram van Ginneken. Ridge-based vessel segmentation in color images of the retina. *IEEE Transactions on Medical Imaging*, 23:501–509, 2004.

[41] Elisa Ricci and Renzo Perfetti. Retinal blood vessel segmentation using line operators and support vector classification. *IEEE Transactions on Medical Imaging*, 26(10):1357–1365, 2007. doi:10.1109/tmi.2007.898551.

[42] D Marín, A Aquino, M E Gegundez-Arias, and J M Bravo. A new supervised method for blood vessel segmentation in retinal images by using gray-level and moment invariants-based features. *IEEE Transactions on Medical Imaging*, 30(1):146–158, 2011. doi:10.1109/tmi.2010.2064333.

[43] Muhammad Moazam Fraz, Paolo Remagnino, Andreas Hoppe, Sergio A. Velastin, Bunyarit Uyyanonvara, and Sarah Barman. A supervised method for retinal blood vessel segmentation using line strength, multiscale gabor and morphological features. *2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pages 410–415, 2011.

[44] Shuangling Wang, Yilong Yin, Guibao Cao, Benzheng Wei, Yuanjie Zheng, and Gongping Yang. Hierarchical retinal blood vessel segmentation based on feature and ensemble learning. *Neurocomputing*, 149:708–717, 2015. doi:10.1016/j.neucom.2014.07.059.

[45] D.T. Mane and U.V. Kulkarni. Visualizing and understanding customized convolutional neural network for recognition of handwritten marathi numerals. *Procedia Computer Science*, 132:1123–1137, 2018. doi:10.1016/j.procs.2018.05.027.

[46] Stéphane Mallat. Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150203, 2016. doi:10.1098/rsta.2015.0203.

[47] Víctor Suarez-Paniagua and Isabel Segura-Bedmar. Evaluation of pooling operations in convolutional architectures for drug-drug interaction extraction. *BMC Bioinformatics*, 19(S8), 2018. doi:10.1186/s12859-018-2195-1.

[48] M. Melinscak, P. Prentasic, and S. Loncaric. Retinal vessel segmentation using deep neural networks. 2015.

[49] H. Fu, Y. Xu, D. W. K. Wong, and J. Liu. Retinal vessel segmentation via deep learning network and fully-connected conditional random fields. *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, pages 698–701, 2016.

[50] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. *International Journal of Computer Vision*, 125(1-3):3–18, 2017. doi:10.1007/s11263-017-1004-z.

[51] Pawel Liskowski and Krzysztof Krawiec. Segmenting retinal blood vessels with newline deep neural networks. *IEEE Transactions on Medical Imaging*, 35(11):2369–2380, 2016.

[52] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net, 2018. URL: https://arxiv.org/abs/1412.6806.

[53] R. Kolar, J. Odstrcilik, J. Jan, and V. Harabis. Illumination correction and contrast equalization in colour fundus images. In *2011 19th European Signal Processing Conference*, pages 298–302, Aug 2011.

[54] Mithilesh Kumar and Ashima Rana. Image enhancement using contrast limited adaptive histogram equalization and wiener filter. *International Journal Of Engineering And Computer Science*, 2016. doi:10.18535/ijecs/v5i6.30.

[55] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. doi:10.1109/5.726791.

[56] Li Deng. A tutorial survey of architectures, algorithms, and applications for deep learning – erratum. *APSIPA Transactions on Signal and Information Processing*, 3, 2014. doi:10.1017/atsip.2014.4.

[57] Sven Arnhold, Sven Hartmann, and Barbara Hammer. *Data restructuring as formal preprocessing for machine learning with neural networks*. 2015.

[58] R Lacroix, F Salehi, XZ Yang, and KM Wade. Effects of data preprocessing on the performance of artificial neural networks for dairy yield prediction and cow culling classification. *Transactions of the ASAE*, 40(3):839–846, 1997.

[59] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[60] Bekir Karlik and A Vehbi Olgac. Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4):111–122, 2011.

[61] A.Dosovitskiy, J.T.Springenberg, M.Riedmiller, and T.Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in Neural Information Processing Systems 27 (NIPS)*, 2014. URL: http://lmb.informatik.uni-freiburg.de/Publications/2014/DB14b.

[62] Yan Wang, Chen Zu, Guangliang Hu, Yong Luo, Zongqing Ma, Kun He, Xi Wu, and Jiliu Zhou. Automatic tumor segmentation with deep convolutional neural networks for radiotherapy applications. *Neural Processing Letters*, 2018. doi:10.1007/s11063-017-9759-3.

[63] Rajpar Suhail Ahmed, Jie Liu, Zhao Fei, and Muhammad Zahid. Automated segmentation of whole cardiac ct images based on deep learning. *International Journal of Advanced Computer Science and Applications*, 9(4), 2018. doi:10.14569/ijacsa.2018.090464.

[64] Lin XiangBo and Li XiaoXi. Image based brain segmentation: From multi-atlas fusion to deep learning. *Current Medical Imaging Reviews*, 14, 2018. doi:10.2174/1573405614666180817125454.

[65] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]). URL: http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a.

[66] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc. URL: http://dl.acm.org/citation.cfm?id=2999134.2999257.

[67] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[68] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2014. URL: http://arxiv.org/abs/1412.6806, arXiv:1412.6806.

[69] Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with Bernoulli approximate variational inference. *arXiv:1506.02158*, 2015.

[70] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001. URL: https://doi.org/10.1023/A:1010933404324, doi:10.1023/A:1010933404324.

[71] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[72] Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 2007. URL: http://dx.doi.org/10.1093/bioinformatics/btm344, arXiv:/oup/backfile/content_public/journal/bioinformatics/23/19/10.1093/bioinformatics/btm344/2/btm344.pdf, doi:10.1093/bioinformatics/btm344.